NO. 14-1401, -1402

IN THE UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT

SAMSUNG ELECTRONICS CORPORATION, LTD.,

Appellant

v.

CCP SYSTEMS AG,

Cross-Appellant

JOINT APPENDIX

From the U.S. Patent and Trademark Office, PTAB Appeal No. 2013-009045, Hon. Howard B. Blankenship, Stanley M. Weinberg, and Stacey G. White, Administrative Patent Judges

---

David L. McCombs
Debra J. McComas
HAYNES AND BOONE LLP
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone: 214.651.5375
Facsimile: 214.200.0525
david.mccombs@haynesboone.com
debbie.mccomas@haynesboone.com

Brian R. Matsui
Jessica E. Palmer
MORRISON & FOERSTER LLP
2000 Pennsylvania Ave., NW
Washington, DC 20006
Telephone: 202.887.8740
Facsimile: 202.887.0763
bmatsui@mofo.com
jpalmer@mofo.com

Mehran Arjomand
MORRISON & FOERSTER LLP
707 Wilshire Blvd.
Los Angeles, CA 90017
Telephone: 213.892.5630
Facsimile: 213.892.5454
marjomand@mofo.com

Attorneys for Appellant Samsung Electronics Corporation, Ltd.

Attorneys for Cross-Appellant, CCP Systems AG

INDEX

Appendix No.	**Date**	**Document**
A0643-655	June 1994	Excerpts of Exhibit D to Request for *Inter Partes* Reexamination: “Interleaf Active Documents” by Paul M. English and Raman Tenneti
A0656-667	11/26/1996	Exhibit E to Request for *Inter Partes* Reexamination: U.S. Patent No. 5,579,519
A0668-676	06/01/1998	Excerpts of Exhibit F to Request for *Inter Partes* Reexamination: “Integrating User Interface Agents with Conventional Applications” by Henry Lieberman
A0742	N/A	Excerpt of Exhibit H to Request for *Inter Partes* Reexamination: Claim Chart for SNQ#2, obviousness based on IBM and Interleaf
A0811	N/A	Excerpt of Exhibit I to Request for *Inter Partes* Reexamination: Claim Chart for SNQs #3 and #4
A0837-838	N/A	Excerpts of Exhibit J to Request for *Inter Partes* Reexamination: Claim Chart for SNQs #5 and #6, anticipation based on Interleaf; and obviousness based on Interleaf and Lieberman
A0973	N/A	Excerpt of Exhibit L to Request for *Inter Partes* Reexamination: Claim Chart for SNQ #9, obviousness based on the Interleaf Patent and IBM
A1008-1009	10/14/2010	Excerpt of Order Granting *Inter Partes* Reexamination
A1024, 1029, 1041, 1047, 1054	11/19/2010	Excerpts of Non-Final Office Action

Appendix No.	**Date**	**Document**
A1074-1220	01/10/2011	Amendment and Response, including: Appendix A and B; and Declaration of Roland Widuch; Declaration of Christoph Picht; Declaration of David Birnbaum.
A1323-1527	03/30/2011	Amendment and Response, including Appendix A and B; and Affidavit of Christoph Picht; Affidavit of David Birnbaum
A2036-2049	01/13/2004	Excerpts of Exhibit S to Third Party Requester Revised Comments: U.S. Patent No. 6,678,705
A2051-2057	04/25/2011	Excerpts of Exhibit T to Third Party Requester Revised Comments: Declaration of Paul Jacobs Under 37 C.F.R. § 1.132
A2115	08/23/2011	Excerpt of Decision on Petitions
A2164-2332	04/09/2012	Excerpts of Action Closing Prosecution
A2382	06/07/2012	Excerpt of Third Party Requester Comments
A2392-2442	07/30/2012	Excerpts of Right of Appeal Notice
A2443-2444	08/27/2012	Notice of Appeal – Owner
A2445	10/29/2012	Excerpt of Patent Owner’s Brief on Appeal Under 35 U.S.C. § 134(c) and 37 C.F.R. § 41.67
A2507	10/29/2012	Excerpt of Appellant Brief – Owner
A2517	11/28/2012	Excerpt of Third Party Requester’s Respondent Brief
A2582-2612	12/18/2013	Record of Oral Hearing
A2613-2614	02/19/2014	Samsung Electronic Corporation’s Notice of Appeal

Appendix No.	Date	Document
A2615-2616	02/19/2014	CCP Systems AG's Notice of Appeal

Dated: December 19, 2014

Respectfully Submitted,

/s/ David L. McCombs

David L. McCombs
Debra J. McComas
HAYNES AND BOONE, LLP
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone: 214.651.5375
Facsimile: 214.200.0525
David.McCombs@haynesboone.com
Debbie.McComas@haynesboone.com

Attorneys For Appellant,
Samsung Electronics Corporation, Ltd.

/s/ Mehran Arjomand

Mehran Arjomand
MORRISON & FOERSTER LLP
707 Wilshire Blvd.
Los Angeles, CA 90017
Telephone: 213.892.5630
Facsimile: 213.892.5454
marjomand@mofo.com

- and -

Brian R. Matsui
Jessica E. Palmer
MORRISON & FOERSTER LLP
2000 Pennsylvania Avenue, NW
Washington, DC 20006
Telephone: 202.887.8740
Facsimile: 202.887.0763
bmatsui@mofo.com
jpalmer@mofo.com

Attorneys for Cross-Appellant
CCP Systems AG

ECF CERTIFICATION

I hereby certify that (i) the required privacy redactions have been made pursuant to Federal Rule of Civil Procedure 5.2; (ii) the electronic submission is an exact copy of the paper document; (iii) the document has been scanned for viruses with the most recent version of a commercial virus scanning program and is free of viruses; and (iv) the paper document will be maintained for three years after the mandate or order closing the case issues.

/s/ Debra J. McComas
Debra J. McComas

D-2320261_1

Form 30

FORM 30. Certificate of Service

UNITED STATES COURT OF APPEALS
FOR THE FEDERAL CIRCUIT

CERTIFICATE OF SERVICE

I certify that I served a copy on counsel of record on Dec 19, 2014
by:

- ☐ US mail
- ☐ Fax
- ☐ Hand
- ☒ Electronic Means (by email or CM/ECF)

David L. McCombs
Name of Counsel

/s/ David L. McCombs
Signature of Counsel

Law Firm: Haynes and Boone, LLP

Address: 2323 Victory Avenue, Suite 700

City, State, ZIP: Dallas, Texas, 75219

Telephone Number: 214.651.5375

FAX Number: 214.200.0525

E-mail Address: david.mccombs@haynesboone.com

NOTE: For attorneys filing documents electronically, the name of the filer under whose log-in and password a document is submitted must be preceded by an "/s/" and typed in the space where the signature would otherwise appear. Graphic and other electronic signatures are discouraged.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
95/001,398	07/16/2010	6684789	P-74637-US	5488

49443 7590 12/20/2013
Pearl Cohen Zedek Latzer, LLP
1500 Broadway
12th Floor
New York, NY 10036

EXAMINER	
STEELMAN, MARY J	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
12/20/2013	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

PTOL-90A (Rev. 04/07)

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SAMSUNG ELECTRONICS CORP. LTD.
Requester and Respondent

v.

CCP SYSTEMS AG
Patent Owner and Appellant

Appeal 2013-009045
Reexamination Control 95/001,398
Patent 6,684,789 B2
Technology Center 3900

Before HOWARD B. BLANKENSHIP, STANLEY M. WEINBERG, and STACEY G. WHITE, *Administrative Patent Judges*.

WHITE, *Administrative Patent Judge*.

DECISION ON APPEAL

I. STATEMENT OF THE CASE

A. Introduction

This is an appeal from the Examiner's decision in an *inter partes* reexamination of U.S. Patent 6,684,789 B2 ("the '789 Patent"), owned by CCP Systems AG ("Appellant"). This reexamination proceeding arose from a third party request filed by Samsung Electronics Corp. ("Requester"). After an Action Closing Prosecution ("ACP"), the Examiner issued a Right of Appeal Notice ("RAN"). The RAN lists claims 1-64, 66-72, and 74-82 as finally rejected. Claims 1-53 are original patent claims. Claims 54-64, 66-72, and 74-82 are proposed new claims. Proposed claims 65 and 73 have been canceled. RAN 2. Patent Owner invokes our review of rejected claims under 35 U.S.C. § 134(b) (2002). An oral hearing was held October 9, 2013. We have jurisdiction under 35 U.S.C. §§ 134(b) and 315 (2002).

We AFFIRM-IN-PART.

B. Related Proceedings

Appellant has informed us that there are no related appeals or interferences before the Board. App. Br. 2. Appellant has also informed us that *CCP Systems AG* v. *Samsung Electronics Corp., Ltd. Samsung Electronics America, Inc., Samsung Networks, Inc. and IBM Corp.,* 2:09-cv-04254 (D. N.J.) is related to this reexamination proceeding. *Id.* 65. The patent portion of that litigation has been stayed. *Id.*

C. The Invention

The '789 Patent is directed to apparatuses and methods to transform a digital print data stream. '789 Patent 1:6-7, Abstract. In the prior art print

systems, an application controls a print driver that converts print data into a format (page description language or "PDL") suitable for the printer. *Id.* 1:9-16, 2:52-61. The invention of the '789 Patent sets out to handle more complex page description language at a higher level of abstraction by parsing an input print data stream for graphically representable objects, storing and transforming these objects, and outputting an output print data stream. *Id.* 2:62-3:20.

Claim 1 is illustrative of the appealed subject matter and is reproduced below:

> 1. A method for the transformation of digital print data streams, in which
>
> (i) an input print data stream (2) is read in,
>
> (ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and
>
> (iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, and
>
> (iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and
>
> (v) the objects thus transformed are combined into an output print data stream (10) and are output,
>
> characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

D. The Prior Art

Pelletier ("Interleaf Patent")	U.S. 5,579,519	Nov. 26, 1996
Berchtold ("'705 Patent")	U.S. 6,678,705 B1	Jan. 13, 2004

AS/400 Guide to Advanced Function Presentation and Print Services Facility, Fourth Edition, IBM Printing Systems, January 1999 ("IBM").

English, Paul M. et al, Interleaf Active Documents, 7(2) Electronic Publishing, 75-87, June 1994 ("Interleaf").

Lieberman, Henry, "Integrating user interface agents with conventional applications," Knowledge-Based Systems 11 (1998), pp 15-23 ("Lieberman").

E. The Rejections

1. Claims 59-64, 66, and 67 stand rejected under 35 U.S.C. § 112, second paragraph, as indefinite. ACP 23-24; RAN 2-6, 44.

2. Claims 64, 66, 67, 72, 74, and 75 stand rejected under 35 U.S.C. § 314(a) as enlarging the scope of the patent being reexamined. ACP 24-26; RAN 6, 44.

3. Claims 1-5, 17, 20-26, 38-41, 54, 56, 59, 61-62, 68, 70, 71, 76-78, and 80-82 stand rejected under 35 U.S.C. § 102(b) as anticipated by IBM. ACP 26-32.

4. Claims 1-7, 11-12, 17, 20-28, 32, 33, 38-44, 48, 49, 54, 56, 59, 61, 62, 68, 70, 71, and 76-82 stand rejected under 35 U.S.C. § 103(a) as obvious over IBM and Interleaf. *Id.* 33-38.

5. Claims 1-14, 16-18, 20-35, 37-51, 53, 54, 56, 57, 59-62, 64, 66, 68-72, 74, and 76-82 stand rejected under 35 U.S.C. § 103(a) as obvious over IBM, Interleaf, and Interleaf Patent. *Id.* 39-49.

6. Claims 15, 19, 36, and 52 stand rejected under 35 U.S.C.

§ 103(a) as obvious over IBM, Interleaf, Interleaf Patent, and Lieberman.[1] *Id.* 50-51.

7. Claims 1, 2, 4-7, 11, 12, 22-23, 25-28, 32, 33, 38, 39, 41-44, 48, and 49 stand rejected under 35 U.S.C. § 102(b) as anticipated by Interleaf. *Id.* 51-56.

8. Claims 1, 2, 4-14, 16-18, 22, 23, 25-35, 37-39, 41-51, 53, 54, 57, 59, 61, 68, 70, and 76-82 stand rejected under 35 U.S.C. § 103(a) as obvious over Interleaf and Interleaf Patent. *Id.* 57-66.

9. Claims 15, 19, 36, 52, and 58 stand rejected under 35 U.S.C. § 103(a) as obvious over Interleaf, Interleaf Patent, and Lieberman. *Id.* 66-67.

10. Claims 1-54, 56-60, 62, 64, 66, 68-71, 76, 77, and 80-82 stand rejected under 35 U.S.C. § 103(a) as obvious over Interleaf Patent and IBM. *Id.* 67-72.

11. Claim 55 stands rejected under 35 U.S.C. § 103(a) as obvious over Interleaf, Interleaf Patent, and ’705 Patent. *Id.* 72-74.

12. Claims 55 and 75 stand rejected under 35 U.S.C. § 103(a) as obvious over IBM, Interleaf, Interleaf Patent, and ’705 Patent. *Id.* 75-77; RAN 32-33.

13. Claims 55, 63, and 67 stand rejected under 35 U.S.C. § 103(a) as obvious over Interleaf Patent, IBM, and ’705 Patent. *Id.* 77-79

[1] Appellant does not separately challenge the rejections directed to dependent claims 15, 19, 36, 52, and 58. App. Br. 5 n 5. Thus, we will not specifically address these claims outside of our conclusion.

F. Issues Presented

The briefing in response to the Examiner's RAN presents us with the following issues:

1. Did the Examiner err in finding indefinite a printer "adapted to transform the graphically representable objects into a format for the control of an output device, to combine the objects into an output print data stream, and to output said combined output print data stream" as recited in claims 59 and 64?

2. Did the Examiner err in concluding that claims 64 and 72 enlarge the scope of the claims of the '789 Patent?

3. Did the Examiner err in finding IBM to disclose (a) analyzing an input print data stream by a parser and (b) assigning a script to an object?

4. Did the Examiner err in finding Interleaf to disclose transforming, reading, and parsing a print data stream?

5. Did the Examiner err in finding the cited references to teach a data stream "formatted in page description language" as recited by claim 76?

6. Did the Examiner err in finding the cited references to teach a "parser [that] is a syntax analyzer" as recited by claim 77?

7. Did the Examiner err in finding the cited references to teach a printer?

8. Did the Examiner err in combining IBM, Interleaf, and Interleaf Patent?

9. Did the Examiner err in combining IBM, Interleaf, Interleaf Patent, and '705 Patent?

10. Did the Examiner err in finding no nexus between the ’789 Patent and Appellant’s evidence of commercial success?

II. ANALYSIS

A. Are Claims 59 and 64 Indefinite?

The Examiner rejected claims 59 and 64 as indefinite because the Specification did not teach a printer adapted to output a print data stream. ACP 24; RAN 5-6. Appellant argues that support for the claimed printer is found in Figure 1 of the ’789 Patent that shows device 3 outputting a print data stream to printer 9. ’789 Patent 8:57-9:12, Fig. 1. Device 3 is “a computer such as a PC or else an intelligent output device such as an intelligent printer – which operates in accordance with the method according to the present invention.” *Id.* 8:59-61. Requester avers the Specification does not describe an embodiment where a printer outputs a print data stream to printer 9. Resp. Br. 3. We disagree. The Specification describes device 3 as either a computer or an intelligent printer in the same passage where it describes device 3 as outputting a print data stream to a device that is “preferably a printer 9.” ’789 Patent 8:59-9:34. Thus, we find that Figure 1 and its supporting text describe an inventive printer that outputs a print data stream. Therefore, we do not sustain the Examiner’s rejection under 35 U.S.C. § 112, second paragraph of claims 59 and 64 and their dependent claims 60-63, and 66-67.

B. Do Claims 64 and 72 Enlarge the Scope of the ’789 Patent Claims?

The Examiner rejected claims 64 and 72 under 35 U.S.C. § 314(a), which provides that “the patent owner shall be permitted to propose … a new claim or claims, except that no proposed … new claim enlarging the

scope of the claims of the patent shall be permitted." 35 U.S.C. § 314(a) (2002). Claims 64 and 72 are directed to "a printing system" that includes a printer or print server. The Examiner found the new claims improper because there are no original claims directed to "a printing system." RAN 6. Appellant argues that original claim 1 includes every limitation found in claims 64 and 72. App. Br. 12. In addition, original claim 17 recites "a system" with a data processing unit programmed to perform the method of claim 1 and original claims 20 and 21 further recite a printer or print server that has a system "as claimed in claim 17." *Id.*; '789 Patent 10:57-62, 11:13-18.

In a reexamination, "[a] new claim enlarges if it includes within its scope any subject matter that would not have infringed the original patent." *Thermalloy, Inc. v. Aavid Eng'g, Inc.,* 121 F.3d 691, 692 (Fed. Cir. 1997) (citing *Quantum Corp. v. Rodime, PLC*, 65 F.3d 1577, 1580 (Fed.Cir.1995)). Requester argues that an accused system performing any of claim 1's steps using logic hardware as opposed to a preprogrammed data processing unit would not fall within the purview of the original claims, but potentially would be captured by new claims 64 and 72. Resp. Br. 4. Appellant avers that the new claims recite a data processing unit and thus, the data processing unit is required to perform all data processing steps. Rebuttal Br. 6.

We agree with Requester. The newly added claims are broader in that a device "adapted to" perform certain steps encompasses more than the originally claimed "data processing unit [that] is programmed" to perform certain steps. The original claims require that the data processing unit

specifically be programmed to carry out the claimed method. Claims 64 and 72, however, have no such requirement. Under the new claims, an accused device with a data processing unit that is not programmed to perform all of the claimed steps could still infringe so long as something else in the device is "adapted to" perform the missing step. Thus, we sustain the Examiner's rejection under 35 U.S.C. § 314(a) of claims 64 and 72 and their dependent claims 66, 67, 74, and 75.

C. Does IBM Disclose Analyzing an Input Print Data Stream by a Parser and Assigning a Script to an Object?

Appellant argues the Examiner erred in determining that the IBM reference anticipates the claims of the '789 Patent. App. Br. 12-13. Appellant asserts that IBM does not disclose an embodiment that both parses an input stream and assigns a script to objects parsed from that input stream as is required by independent claims 1, 22, 38, 54, and 68. *Id.* 13-14.

According to Appellant, IBM discloses two embodiments. *Id.* 13. In the first embodiment, an application program describes the formatting of the document to be printed ("program-described"). *Id.* (citing IBM 25). In the second embodiment, an application outputs raw data with the formatting to be "externally-described" by a data description specification ("DDS"). *Id.* Appellant argues that the Examiner found DDS to disclose the claimed script, but that DDS is never applied to "program-described" output. Thus, the Examiner improperly relied on parsing found in the "program-described" embodiment and a script found in the "externally-described" embodiment. Requester asserts that Appellant is taking a narrow view of IBM's

disclosures and that the "program-described" and "externally-described" functionalities are not mutually exclusive. Resp. Br. 5.

IBM describes "DDS as an enabler for document and reports … [that] provides the application programmer with the capability to produce very customized output," but notes that "there are environments where this tight integration is less desirable … [because] it can make the task of coding application logic and output logic more complex because the logic is intertwined." IBM 193. IBM then describes "[n]ew output formatting objects …, page definitions and form definitions" that "provide a means to separate page formatting from the application program." *Id.*

The Examiner points to IBM's discussion of splitting output into "print fields" as disclosing the claimed parser. ACP 27 (citing IBM 194). That portion of IBM describes using a page definition so that "[i]nput print lines are read in, optionally parsed into individual fields, and place[d] on the page." IBM 194. However, "page definition is an alternative to DDS for formatting data on a page independent of the application program." IBM 37. Thus, Appellant argues that page definitions and DDS are distinct alternative embodiments. Requester asserts that DDS scripts may be applied to the parsed data through the application of an overlay as shown in Figure 116 of IBM. Resp. Br. 5 (annotated Fig. 116). This, however, is not supported by the reference. IBM describes using form definitions in conjunction with page definitions to supply and place an overlay on the page. IBM 195, 214. We have not been directed to persuasive evidence that DDS is ever applied to data that has been parsed with a page definition. Thus, we do not sustain

the Examiner's rejection under 35 U.S.C. § 102 of claims 1-5, 17, 20-26, 38-41, 54, 56, 59, 61, 62, 68, 70, 71, 76-78, and 80-82 as anticipated by IBM.

D. Does Interleaf Disclose Transforming, Reading, and Parsing a Print Data Stream?

Appellant asserts that the Examiner erred in concluding that Interleaf anticipates the claims of the '789 Patent. Specifically, Appellant argues that Interleaf does not disclose transforming, reading, or parsing a print data stream as required by independent claims 1, 22, 38, 54, and 68. App. Br. 24.

Interleaf is "[a] commercial structured document processing system" for "the design, implementation, and delivery of active documents." Interleaf 75. Interleaf's documents include component objects that contain content and document formatting. *Id.* 76. One of Interleaf's purported advantages is its ability to use external publishing functionality including document printing. *Id.* 82. In Interleaf, "active objects within the document file stream become activated … when the document is opened programmatically or by the user for viewing, editing, or printing." *Id.* 84.

Appellant argues that Interleaf does not disclose a print data stream. App. Br. 24. The Examiner construed "digital print data stream" to include on "a 'digital' 'data stream' where the stream may be sourced from a 'document' and may or may not be output to a 'printer.'" ACP 148. The Specification does not define this term and we see no error in the Examiner's reasonable construction. Pursuant to this construction, the Examiner found that Interleaf's "document file stream" discloses the claimed "digital print data stream." *Id.* The Examiner found that Interleaf's document file stream, which is sourced from an Interleaf document, may be printed and that it is

parsed for active objects. *Id.* 51. Dr. David Birnbaum testified on behalf of Appellant that Interleaf's file stream is merely what is read in when an Interleaf document is opened and that while Interleaf conceivably may create a print data stream no such print data stream is discussed in the reference. Birnbaum Decl. ¶¶ 100-03. According to Dr. Birnbarum, a print data stream would not exist until after a user clicks a print icon. *Id.* ¶ 102. We are not persuaded by Dr. Birnbaum's testimony because it fails to provide convincing evidence that an Interleaf document file stream becomes a distinct print data stream once a user presses a print button. We agree with the Examiner's reasonable findings and we concur that Interleaf's "document file stream" falls within the broadest reasonable interpretation of the claimed "print data stream."

The remaining question is whether Interleaf discloses reading, parsing, and transforming a print data stream. As to the transforming step, the Examiner points to Interleaf's discussion of changing a document's style in the middle of a text stream, changing objects in the stream, and programmatically applying different graphics properties such as changes to color, size, or position of an object. ACP 52 (citing Interleaf 76, 81-84). Appellant argues that these disclosures refer to editing or creating a document and not printing the document. App. Br. 25. We disagree with Appellant. As noted above, the print data stream as broadly construed includes Interleaf's document file stream. Thus, we agree with the Examiner's finding that Interleaf discloses transforming a print data stream.

As to the "reading in" step, the Examiner found this to be disclosed by Interleaf's discussion of opening a document for viewing, editing, or

printing. ACP 51-52. Appellant argues that this is akin to opening a document in Word and is completely unrelated to the claimed reading in of a print data stream. App. Br. 26. Here again, Appellant's view of a print data stream is too narrow. We agree with the Examiner's finding that Interleaf's discussion of opening a document discloses the claimed "reading in" of a print data stream.

Finally, as to the parsing step the Examiner looks to Interleaf's discussion of a Lisp interpreter that analyzes the data stream by means of a parser. ACP 52. Appellant argues that the Lisp interpreter only interprets Lisp scripts and "it is incapable of interpreting print data streams." App. Br. 27; Birnbaum Decl. ¶ 105. The Examiner, however, points out that Interleaf discloses binding the Lisp code and the underlying document data into a single document file stream. ACP 52; *see also* Interleaf 77 ("Lisp scripts can be attached to any Interleaf object, stored within or external to a document."). As such, the Lisp interpreter operates on the Interleaf document file stream. Thus, we are persuaded that Interleaf discloses parsing of a print data stream. Therefore, we a sustain the Examiner's rejection under 35 U.S.C. § 102 of claims 1, 2, 4-7, 11, 12, 22, 23, 25-28, 32-33, 38-39, 41-44, 48, and 49 as anticipated by Interleaf.

E. Do the Cited References Teach a Data Stream "Formatted in Page Description Language"?

Claim 76 depends from claim 1 and further limits the input data stream by requiring it to be "formatted in a page description language." The Examiner found this limitation to be taught by IBM's disclosure of Postscript files and also by the disclosure of Interleaf frames found in both

Interleaf and Interleaf Patent. ACP 31, 64. Appellant disagrees with both findings. App. Br. 20, 27.

As to the IBM reference, Appellant argues that the Examiner did not examine this limitation in the broader context of the claim. App. Br. 20. In particular, claim 1 requires that an input data stream be assigned a script. The Examiner's rejection would apply DDS to the Postscript file, however, "a Postscript file would *already* be formatted and not require or allow a DDS file to format it." *Id.* (citing IBM 25). IBM states that "document data streams, such as Postscript … can be converted to AFP and then stored or printed from the AS/400." IBM 8. In addition, IBM teaches that an AFP file can contain DDS keywords and that AFP functions include printing graphical information from DDS. Resp. Br. 9; IBM 26. Thus, Requester argues that IBM teaches applying DDS to a Postscript file. Resp. Br. 9. We agree with Requester and find that the Examiner did not err in finding that one of ordinary skill in the art would read the IBM reference to teach the limitations of claim 76.

As to Interleaf and Interleaf Patent, Appellant argues that the Examiner erred because "Interleaf frames are merely the manner in which objects are depicted on a screen; they have nothing to do with printing, and certainly do not constitute an input print data stream." App. Br. 27. As previously discussed, print data streams encompass the Interleaf's document file stream. *Supra* pp. 11-12. Frames are a component of that document file stream. Interleaf 76. Thus, the Examiner did not err in finding the Interleaf references to teach the limitations of claim 76.

F. Do the Cited References Teach a "Parser [That] Is a Syntax Analyzer"?

Claim 77 depends from claim 76 and further requires the parser to be a syntax analyzer. The Examiner found this limitation to be taught by IBM's disclosure of assigning a DDS script to an object and by the Lisp interpreter disclosed in the Interleaf references. ACP 31, 64. As to the IBM reference, Appellant argues that DDS is not applied to a Postscript file. App. Br. 20. We disagree for reasons discussed above. *Supra* p. 12. As to the Interleaf references, the Appellant argues that the Lisp interpreter analyzes the Lisp scripts and not the input data stream. We disagree for reasons previously discussed. *Supra* p. 13. Thus, the Examiner did not err in finding that IBM and the Interleaf references teach the limitations of claim 77.

G. Do the Cited References Teach a Printer?

Claims 20, 59, and 64 each are directed to a printer. These claims stand rejected over IBM's teachings as combined with the teachings of Interleaf and/or Interleaf Patent. Appellant argues that neither IBM nor the Interleaf references teach a printer. App. Br. 21-22, 28. Appellant contends that the AS/400 described in the IBM reference is not a printer; instead it merely communicates with printers. App. Br. 21. Further, Appellant maintains that being attached to a printer via a cable or network does not make as the AS/400 a printer nor does it include a printer. *Id.* 22. Requester argues that the claims do not require a single chassis and thus, IBM's disclosure of an AS/400 connected to a printer teaches the recited printers. Resp. Br. 9 (citing RAN 11). Appellant responds that even under the

broadest reasonable interpretation a printer must be a single device and not a distributed system. Rebuttal Br. 17; *see also* App. Br. 22 n. 14. We disagree. Appellant has not directed us to persuasive evidence in the Specification that would limit the recited printer to a single chassis. Thus, we see no error in the Examiner's finding that IBM teaches a printer.

Claim 59, which is directed to a printer, stands rejected as obvious over the Interleaf references. The Examiner found that "the Interleaf patent is directed towards 'a Document Processing System for creating, editing, printing and presenting active electronic documents.' Thus, the 'system' of the Interleaf patent is for 'printing,' and can therefore be considered a 'printer.'" RAN 10. Appellant argues that this is insufficient to render obvious the claimed printer. App. Br. 28. We agree. The Examiner has not provided sufficient evidence that one of ordinary skill in the art would learn the claimed printer from the Interleaf references. Those references merely state that Interleaf is "for printing" without any disclosure as to a printer. Thus, the Examiner erred in rejecting claim 59 under 35 U.S.C. § 103 over Interleaf and Interleaf Patent.

H. Did The Examiner Err in Combining IBM, Interleaf, and Interleaf Patent?

Appellant argues that the Examiner erred in combining the teachings of IBM, Interleaf, and Interleaf Patent. App. Br. 29-34. Appellant asserts that "the Examiner defined the field of the invention broadly, inconsistently, and artificially, and then tried to shoehorn the prior art into it." *Id.* 31. According to Appellant the Examiner found that the '789 Patent's field of art to be "object-oriented document publishing systems with scripting

functionality." *Id.* (citing ACP 33). This, however, misstates the Examiner's finding. The Examiner was discussing the analogous nature of IBM and the Interleaf references. ACP 33, 69. According to the Examiner, it would have been obvious to modify IBM with the teaching of Interleaf as related to assigning a script to be executed. *Id.* 33, 69-70. The Examiner further found that "Interleaf suggests the combination because it is explicitly designed to leverage outside printing and publishing functionality." *Id.* 34.

Appellant also contends that the Examiner was incorrect in finding that "documents produced by IBM are the same type of documents that the Interleaf Patent's EDPS processes and prints." App. Br. 32 (citing ACP 70). Appellant states, without citation to the reference, that Interleaf's active documents must be converted to inactive documents before they may be printed. *Id.* 32. Requester argues that this is contrary to the record because in Interleaf, "active objects within the document file stream become activated … when the document is opened programmatically or by the user for viewing, editing, or printing." Resp. Br. 12 (quoting Interleaf 84).

The Examiner found that the combination of IBM's printing process and Interleaf Patent's print-scripting was a combination of well-known techniques to yield predictable results. ACP 69; *see also* ACP 34-35 (making same argument in regards to the combination of Interleaf and IBM). Our reviewing court has reaffirmed that "[t]he combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results." *Leapfrog Enter., Inc. v. Fisher-Price, Inc.*, 485 F.3d 1157, 1161 (Fed. Cir. 2007) (quoting *KSR Int'l v. Teleflex Inc.,* 550 U.S. 398, 416 (2007)). We find the Examiner's reasoning

to be rational and well supported by the factual record. Accordingly, we find Appellant's arguments unpersuasive that the cited references have been improperly combined by the Examiner.

I. Does the '705 Patent Teach Receiving Email Via a Script ?

Claims 55, 63, 67, and 75 each recite a device adapted or programmed "to send and receive e-mails in the cases defined in the script." The Examiner cites the '705 Patent as teaching receiving email messages via a script. RAN 32. Appellant argues that this limitation is not taught by the '705 reference. App. Br. 40-41. Requester asserts that the '705 Patent shows that the disputed limitation was known in the art. Resp. Br. 14. We agree with Appellant. The '705 Patent does not teach receiving an email via script; instead the cited portions of the '705 Patent describe a user emailing a document to a server and the server running scripts to parse the message and store the contents of the email to media. *Id.* 3:17-22, 40-50. Thus, this reference does not teach "receiv[ing] e-mails in the cases defined in the script" as recited by claims 55, 63, 67, and 75. Therefore, we do not sustain the Examiner's rejections of claims 55, 63, 67, and 75 under 35 U.S.C. § 103.

J. Is There a Nexus Between the '789 Patent and Appellant's Evidence of Commercial Success?

Appellant argues that its evidence of commercial success outweighs any prima facie case of obviousness. App. Br. 35-36. The Examiner disagreed and found there to be no nexus between Appellant's evidence and the '789 Patent. RAN 20.

Appellant provides declarations from its CEO, Roland Widuch, and Director of Business Development, Christoph Picht. Widuch testifies that IBM took an exclusive license to Appellant's JScribe® software. Widuch Decl. ¶ 7. The term "JScribe" is listed on the face of the '789 Patent. '789 Patent 5:56-59 ("systems operating by the method according to the invention are also designated JScribe (registered trademark) systems and, accordingly, the method according to the invention is also designated JScribe."); *see also* 6:38-41, 6:66-67, 7:57-59, 8:26-34. Picht testifies that JScribe® embodies at least claims 1-15 of the '789 Patent. Picht Decl. ¶¶ 11, 15. Widuch further testifies that IBM provided a sublicense to Konica-Minolta Business Solutions, Inc. Widuch Decl. ¶ 8. In addition, Widuch states that Requester took a sublicense and agreed to pay IBM a lump sum and a running royalty for each device that includes JScribe® technology. *Id.* ¶ 10. IBM and Requester issued a joint press release announcing the launch of printers and multifunction products using JScribe®. *Id.* ¶ 11. Requester issued a press release and a white paper discussing its use of JScribe®. *Id.*¶¶ 15-16.

Requester argues that Appellant's evidence lacks the required nexus. Resp. Br. 13. Requester asserts that the licenses in question do not specifically mention the '789 Patent. *Id.* In addition, the licenses were for the source code of JScribe® and source code provides a value and a commercial benefit in itself. *Id.* Further, Requester avers that the Picht declaration is conclusory. *Id.* We agree with Requester. Picht's declaration is conclusory in that it merely repeats the limitations of the claim without providing further analysis as to JScribe's® functionality. In addition, none of the licenses relied upon by Appellant are of record and the testimony

provided regarding these licenses fails to state that the ’789 Patent is specifically mentioned in any of the agreements. Thus, we find that the Examiner did not err in finding there to be a lack of nexus between Appellant’s evidence of commercial success and the claims of the ’789 Patent.

III. CONCLUSION

To summarize, our decision is as follows.

- The Examiner’s rejection of claims 59-64, 66, and 67 under 35 U.S.C. § 112, second paragraph is reversed.
- The Examiner’s rejection of claims 64, 66, 67, 72, 74, and 75 under 35 U.S.C. § 314(a) is affirmed.
- The Examiner’s rejection of claims 1-5, 17, 20-26, 38-41, 54, 56, 59, 61-62, 68, 70, 71, 76-78, and 80-82 under 35 U.S.C. § 102(b) as anticipated by IBM is reversed.
- The Examiner’s rejection of claims 1-7, 11-12, 17, 20-28, 32, 33, 38-44, 48, 49, 54, 56, 59, 61, 62, 68, 70, 71, and 76-82 under 35 U.S.C. § 103(a) as obvious over IBM and Interleaf is affirmed.
- The Examiner’s rejection of claims 1-14, 16-18, 20-35, 37-51, 53, 54, 56, 57, 59-62, 64, 66, 68-72, 74, and 76-82 under 35 U.S.C. § 103(a) as obvious over IBM, Interleaf, and Interleaf Patent is affirmed.
- The Examiner’s rejection of claims 15, 19, 36, and 52 under 35 U.S.C. § 103(a) as obvious over IBM, Interleaf, Interleaf Patent, and Lieberman is affirmed.

- The Examiner's rejection of claims 1, 2, 4-7, 11, 12, 22-23, 25-28, 32, 33, 38, 39, 41-44, 48, and 49 under 35 U.S.C. § 102(b) as anticipated by Interleaf is affirmed.
- The Examiner's rejection of claims 1, 2, 4-14, 16-18, 22, 23, 25-35, 37-39, 41-51, 53, 54, 57, 61, 68, 70, and 76-82 under 35 U.S.C. § 103(a) as obvious over Interleaf and Interleaf Patent is affirmed.
- The Examiner's rejection of claim 59 under 35 U.S.C. § 103(a) as obvious over Interleaf and Interleaf Patent is reversed.
- The Examiner's rejection of claims 15, 19, 36, 52, and 58 under 35 U.S.C. § 103(a) as obvious over Interleaf, Interleaf Patent, and Lieberman is affirmed.
- The Examiner's rejection of claims 1-54, 56-60, 62, 64, 66, 68-71, 76, 77, and 80-82 under 35 U.S.C. § 103(a) as obvious over Interleaf Patent and IBM is affirmed.
- The Examiner's rejection of claim 55 under 35 U.S.C. § 103(a) as obvious over Interleaf, Interleaf Patent, and '705 Patent is reversed.
- The Examiner's rejection of claims 55 and 75 under 35 U.S.C. § 103(a) as obvious over IBM, Interleaf, Interleaf Patent, and '705 Patent is reversed.
- The Examiner's rejection of claims 55, 63, and 67 under 35 U.S.C. § 103(a) as obvious over Interleaf Patent, IBM, and '705 Patent is reversed.

IV. DECISION

The Examiner's decision rejecting claims 1-54, 56-62, 64, 66-72, and 74-82 is affirmed. The Examiner's decision rejecting claims 55 and 63 is reversed.

Requests for extensions of time in this *inter partes* reexamination proceeding are governed by 37 C.F.R. § 1.956. *See* 37 C.F.R. § 41.79.

AFFIRMED-IN-PART

PATENT OWNER:

PEARL COHEN ZEDEK LATZER, LLP
1500 BROADWAY, 12TH FLOOR
NEW YORK, NEW YORK 10036
TEL: (646) 878-0800
FAX: (646) 878-0801

THIRD PARTY REQUESTER:

HAYNES AND BOONE, LLP
IP SECTION, 2323 VICTORY AVENUE, SUITE 700
DALLAS, TEXAS 75219
TELEPHONE: 214/651-5533
FACSIMILE: 214/200-0853
ATTORNEY DOCKET: 38512.24

US006684789B2

(12) **United States Patent**
Krautter

(10) **Patent No.: US 6,684,789 B2**
(45) **Date of Patent: Feb. 3, 2004**

(54) **METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER**

(75) Inventor: **Thomas Erfinders Krautter**, Stuttgart (DE)

(73) Assignee: **CCP Systems AG**, Stuttgart (DE)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **10/275,784**

(22) PCT Filed: **May 11, 2001**

(86) PCT No.: **PCT/DE01/01796**

§ 371 (c)(1),
(2), (4) Date: **Nov. 7, 2002**

(87) PCT Pub. No.: **WO01/88840**

PCT Pub. Date: **Nov. 22, 2001**

(65) **Prior Publication Data**

US 2003/0140809 A1 Jul. 31, 2003

(30) **Foreign Application Priority Data**

May 17, 2000 (DE) 100 24 177
Jan. 26, 2001 (DE) 101 03 733

(51) **Int. Cl.**[7] **B41F 1/54**
(52) **U.S. Cl.** **101/484**; 101/486; 400/61; 400/62
(58) **Field of Search** 101/484, 485, 101/486; 400/61, 62, 63, 76; 358/1.1, 1.9, 1.15, 1.16, 1.17, 1.18

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,216,754 A		6/1993	Sathi et al.	
5,566,278 A	*	10/1996	Patel et al.	358/1.15
6,006,013 A		12/1999	Rumph et al.	

FOREIGN PATENT DOCUMENTS

EP	0 109 615 B1	6/1994
EP	0 964 339 A2	12/1999
EP	1 061 456 A2	12/2000
GB	2 357 348 A	6/2001
WO	WO-00/17748	3/2000
WO	WO-00/55720	9/2000

* cited by examiner

Primary Examiner—Andrew H. Hirshfeld
Assistant Examiner—Minh Chau
(74) *Attorney, Agent, or Firm*—Squire, Sanders & Dempsey L.L.P.

(57) **ABSTRACT**

A method for the transformation of digital print data streams, in which an input print data stream (**2**) is read in, this is analyzed by means of a parser (**4**) for graphically representable objects (**5**, **5***a*) and is split up into these graphically representable objects (**5**, **5***a*), and the graphically representable objects (**5**, **5***a*) are stored in a memory (**6**) in an object-oriented format, and the graphically representable objects (**5**, **5***a*) stored in the memory (**6**) in an object-oriented format are transformed into a format for the control of an output device (**9**), preferably a printer, and the objects thus transformed are combined into an output print data stream (**10**) and are output, graphically representable objects (**5**, **5***a*) being stored in the memory (**6**) in an object-oriented format, to which at least one stored script (**5***a*) is assigned, which is executed in the cases defined in the script (**5***a*).

53 Claims, 1 Drawing Sheet

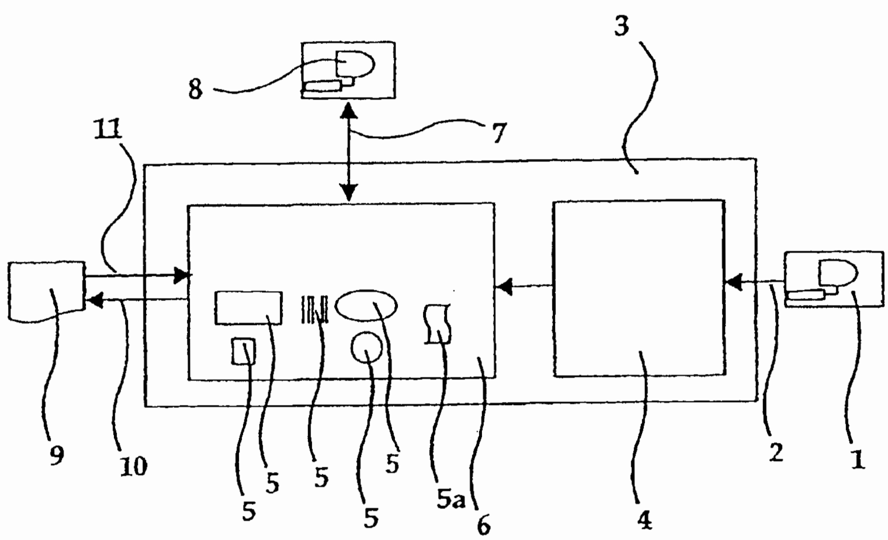
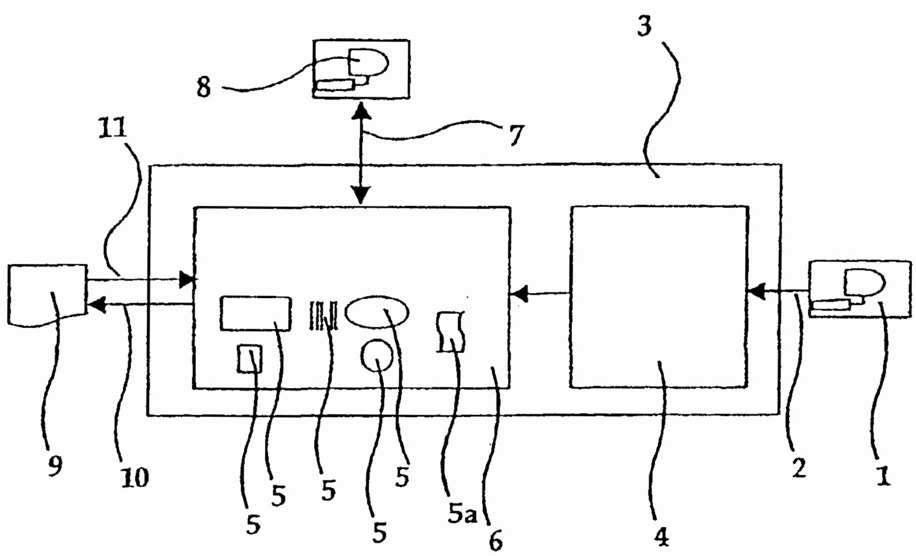


FIG. 1



METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER

The present invention relates to a method and system for the transformation of digital print data streams and corresponding printer and printer server.

Virtually all the output devices which are common nowadays use "page description languages", also called PDL, to produce printed documents. Here, an application program controls a driver for the output device (for example a printer driver). This driver converts information about the graphic objects to be output—for example text or image information—into the respective PDL suitable for the printer used, so that the latter can hereby be controlled directly.

More recent output devices, such as laser printers or digital color printers for example, also offer the possibility of buffering the data streams coming in to control them and, for example, using them as an original form for further incoming print data. This makes it possible to dispense with forms needed for the respective printing, such as letter paper, invoice forms or the like for example, in the individual case. Instead, the application software respectively used merely calls up the form stored once in the printer and combines it with the current print data. In this way, the accumulation of data, for example in networks, can be reduced considerably. However, the result is also organizational advantages: since the forms used no longer have to be kept in reserve by each individual user on his computer, in this way standardized use forms can be achieved, which firstly helps to ensure the often desired standard appearance of a company or an institution and secondly also makes it easier to use current form versions.

However, these aforementioned advantages are normally not used, since the printers used in a company or an institution—with regard to their control—are often not uniform and therefore the use of the functions described above is too complicated, since the appropriate forms either have to be available for each printer model used, which would be very labor-intensive, or only specific printers can be used for specific applications, which is very inflexible.

One possibility of solving this problem is to circumvent the abovedescribed inhomogeneity of the output devices used by employing methods for the conversion of various data stream formats for controlling output devices, which makes it possible for all the computers which produce print data streams to be output to use a standard format for this purpose, by each printer being assigned an interface—be it a dedicated device, be it merely in the form of a software filter—which makes use of such a method and, on the side of the input data stream, uses the format to be used uniformly and, on the side of the output data stream, uses the specific format of the printer to be controlled.

Such a solution is described, for example, by EP 0 109 615 B1, which refers to a method for the conversion of text which is represented in the form of digital data. However, the method taught by this document has considerable disadvantages with regard to the possibilities of current systems from information technology: for example, the method is suitable only for those input print data streams which, in their syntax, follow a format description language whose syntax may be described with the aid of "regular expressions". This is because the method taught in EP 0 109 615 B1 makes use of a status machine, implemented by means of "key status variables", for the recognition and conversion of input control objects recognized in the input print data

stream into output control objects. These output control objects are in this case produced directly from the input control objects—specifically in accordance with a fixed assignment—as a function of the respective state representing the key status variables. Such a procedure corresponds to the functioning of the theoretical model of the Moore or Mealy machines, which operate quite efficiently but permit only, the recognition of regular expressions. For these circumstances surrounding information technology at the priority date of EP 0 109 615, such a simple possible transformation may have been sufficient, since—as can already be gathered from claim 1 there—only text had to be converted, apart from format information.

For the current circumstances of PDLs or else other input formats to be recognized where possible, such as HTML or XML, this no longer applies in any way, however. In the meantime, these have been built up in such a complex way with regard to their possibilities that a status machine is no longer in any way adequate for their recognition and conversion.

However, the target format, into which the print data stream is to be transformed, nowadays places high requirements on a transformation: although in principle there would be the possibility here likewise of using the smallest common multiple of the functions of current printing format and in this way of reducing the effort on transformation, this convenience in the design of the transformation process would be brought at great expense in the operation of the method, since in this way the accumulation of data in networks would be increased again, since powerful printer control possibilities which as a rule become more and more specific with regard to the printer type used as the complexity increases, would necessarily have to be dispensed with. Such an increased accumulation of data would, however, again stand in the way of the objective of reducing the data traffic in the network by using PDLs. Thus, at the same time, there is a requirement on the transformation process that the latter produces the preconditions that the target formats can be produced in the most flexible manner possible with all their available printing functions, in order that the traffic on the data transmission lines can thus be minimized.

Furthermore, it is necessary to state that printing systems, even today, still only fulfill a single purpose: namely printing. All the manufacturers of laser printers and digital copying systems have made great efforts in recent years to match the processor powers, storage capacities and additional options (such as memory cards, hard drives, network cards) of these systems to the increasing requirements. However, the manner in which printers and copiers are controlled and programmed has not changed significantly in the last ten years.

Printing systems are still controlled by a page description language (PDL) such as PCL, Postscript or Prescribe. It permits a document and its components to be described adequately. However, the many additional options of modem printing and copying systems available in the meantime cannot be used. The consequence of this is that, even today, the entire printing process is controlled and monitored by a host computer. Its task substantially comprises converting the respective information exactly into the page description language "understood" by the printing system.

It is therefore an object of the present invention to specify a method for the transformation of digital print data streams which is both capable of recognizing more complex page description languages, whose syntax may no longer be described with the aid of simple regular expressions, and also provides the preconditions that the recognized graphic

objects can be transformed into a target format, but also processed further, as flexibly and effectively as possible, that is to say with regard to their description at the highest possible level of abstraction.

According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in, this is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects, and the graphically representable objects are stored in a memory in an object-oriented format, and the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer, and the objects thus transformed are combined into an output print data stream and are output, and which, according to the invention, is characterized in that graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

In this case, as opposed to the use of a status machine, the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used. Instead, such a parser, in terms of its theoretical performance, corresponds to a Turing machine and therefore ensures the theoretically maximum achievable performance for the analysis and splitting up of formal languages.

Furthermore, storing the graphically representable objects—and therefore of course also the scripts, which themselves are certainly also graphically representable objects—in a memory in an object-oriented format achieves the situation where the objects recognized by the parser are then available in this intermediate format which is extremely beneficial for further processing.

The objects are preferably managed here by means of a "display list management", which supports one-page and multi-page documents at as many levels as desired and which can be expanded dynamically by new objects. The individual graphic objects are stored by using their membership of specific—expediently suitably hierarchically organized—classes such as relating to the class of points, ellipses, circles, lines, polygons, rectangles, squares or else to the more complex object types, such as bar codes, more complex texts or freely definable elements such as color profiles or fonts, which permits their effective conversion into an output print data stream, since, through the class of the respective object, there is already implicit information available about its possible transformation into the format of the output print data stream. For example, via an object of the type square, it is already known from the object hierarchy that this is a subclass of the rectangle. If, then, the target format for which an output print data stream is to be produced provides speech constructs relating to the description of rectangles in the page description language, then it is clear, merely on the basis of the position of the square in the object class hierarchy, that this is also a rectangle—albeit with special characteristics—and to this extent the possibilities of the target format with regard to rectangles can also be used for an object in the square class.

In addition to such implicit information—which can be derived from the object class hierarchy—about the individual objects, however, it is also possible to add to the objects explicit information about their possible conversions into specific target formats, it being advantageously possible for this also to be combined with the abovedescribed implicitly provided information, for example by a conversion method into a specific target format being added to a class which is arranged higher in the object class hierarchy, and then automatically also being available to the objects of subordinate, lower-ranking classes by way of inheritance, if a better specified method is not already assigned to said subordinate classes.

In one embodiment of the method according to the invention, the graphically representable objects are combined into super-objects of higher complexity before being stored in the memory.

The super-objects obtained in this way are then stored in the memory in the object-oriented format. In this way, less complex graphic objects can be combined to form more complex graphic super-objects. For example, sequences of lines which in each case join one another at the ends and have been recognized as graphic objects in the input print data stream can be combined to form a graphic polygon super-object. Such a combination offers various advantages, such as easier handling of the super-object stored as a whole as compared with the individual objects, since said super-object can then be treated uniformly by the methods for the super-object class with effect for all the part objects combined in it. It also helps, in certain circumstances, to further minimize the data traffic on the transmission lines used, since an object once combined is subsequently also forwarded in combined form in the output print data stream—if technically supported there—which generally requires a lower data volume to be transmitted than the transmission of the individual objects.

A preferred embodiment of the method according to the present invention is characterized in that a parser is used for the analysis and splitting up into the graphically representable objects, which, in the theoretical model, corresponds to an automatic push-down facility and which is therefore capable of analyzing and splitting up languages with "context-free grammars" particularly effectively.

A further embodiment of the method according to the present invention is characterized in that feedback messages referring to the output print data stream output are read in and are analyzed for error messages which indicate that the output device, preferably the printer, has recognized a transformed graphic object in the output print data stream which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device, are slipped into the output print data stream which is output to the output device.

In this way, it is likewise possible to test whether the driven output device is, for example, capable of recognizing and outputting a bar-code object directly or not. If it is not capable of this and reports this back, then the bar code is simply split up into objects of the next lower hierarchy, for example filled rectangles, and a further try is made with these objects. This is continued until—if necessary until the graphic objects are split up into individual points—the output attempt is successful. The object-oriented data structure with its object hierarchy, chosen for the intermediate format, also proves to be particularly suitable for this procedure. For the further performance of the method, it is preferably noted at which level of the object classes in each case the splitting process was successful for a specific output device, in order then, in the next attempt, already to begin the output process at this level, in order also thus to avoid unnecessary data transfers, but likewise to utilize the maximum level of abstraction of the output device. In this way,

the data volume to be transmitted is reduced to the necessary extent, even with high flexibility.

In an embodiment of the method according to the present invention, at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment, which permits the incorporation of all the devices needed in the widest sense for document processing.

A further preferred embodiment of the method according to the invention is characterized in that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

The script automatically receiving data can preferably also request this data automatically.

It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver.

It can also in turn reassign the data received by it to the graphic object associated with it, and forwards the graphic object associated with itself to a receiver together with the data requested, received and reassigned by itself, or else print out said data.

In relation to the above explanations, it should be noted that the embodiments of the method according to the invention which themselves provide other objects with objects, for example by forwarding them or keeping them ready to receive or for interrogation by a script, for example, are also covered by the term "dynamic object linking" (DOL).

Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails and of printing original print and image data without a printer driver. They are able to store any desired information on hard disks or memory cards and make said data available to all the devices connected in the network and Internet. In other words, they independently undertake demanding tasks in information processing and provision, in order to relieve host computers and personal computers of quite a lot of administrative tasks. In a heterogeneous network and printing environment with laser printing and copying systems from different manufacturers in combination with impact printers and special printing systems, they also make it possible to administer all the connected printing systems with the aid of a single standardized programming language, namely the script language, and therefore reduce the effort on administration to a minimum. At this point, it should be mentioned that these systems operating by the method according to the invention are also designated JScribe (registered trademark) systems and, accordingly, the method according to the invention is also designated JScribe (registered trademark).

When JScribe (registered trademark) is used, developers and system houses will therefore be in a position to provide objects and functions which are stored in resident form in the printing system and permit and control desired individual operating sequences. These objects and functions can use any functionality provided by the JScribe (registered trademark) basic technology, including extremely demanding commands for the job or page processing and for the complete control of the print data and emulations. The method according to the invention preferably also enables access to internal printer functions and status information (page counter, network components, file system and so on), for example via a script.

The method according to the invention is preferably characterized in that graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the case of the output of the object defined in the script. In this way, for example, it is possible to execute such scripts, for example Visual Basic Scripts, Java Scripts or else "stream code" in an event-oriented manner, for example in the case where a form object is printed out, likewise "ON-PRINT" by which means, for example, to execute such functions as the printing of copies of the same form with the same net data but on different paper from different trays. In particular in interaction with those embodiments of the method according to the invention which control external devices, such as folding or enveloping machines or else stapling machines, this is particularly advantageous.

However, it may also be the case that at least one case relating to the execution of the script is defined in the respective script and occurs automatically, preferably without further influence from outside.

For example, the automatically occurring case, defined at least in the respective script, relating to the execution of the script can be configured as a timer, that is to say as a case which occurs automatically as a result of the expiry of a time, this timer preferably operating cyclically, that is to say starting itself again upon expiry.

Automatic scripts can therefore intrinsically become active and, for example, load the daily newspaper, where possible itself assembled from different sources, from the Internet, assign the found, loaded and analyzed information to a stored object and then print this object, completely without the participation of a PC or other host computer to which the printer would be connected.

For example, the simple download of JScribe (registered trademark) sequences (scripts with appropriately associated objects) can, for example, arrange for the printer automatically to fetch information about current share prices, to format it and to print it out. Image information, text documents, web pages, XML documents and any other desired print data can be analyzed while dispensing with any preparation by the PC (for example by a printer driver), modified if necessary and printed out in optimum quality. Since JScribe (registered trademark) can also be employed simultaneously as a server version for computer systems, printing systems are for the first time made capable of accessing stocks of data on host systems (for example SQL databases) interactively during the printing operation.

The language used for the scripts according to the present invention is preferably Java Script. Java Script, as a world-established standard for the script-controlled, intelligent programming of web pages, has triggered in the Internet an avalanche of innovative and functional solutions which have contributed decisively to ringing in the age of eBusiness and eCommerce. This intelligent technology, which has so decisively marked the worldwide, rapid development of the Internet, is therefore now also available for printing systems for the first time and here preferably forms the basic technology for script applications in the area of the present invention, and consequently print and document management, which is certainly uniquely and, as compared with established solutions, considerably more cost-effective.

With JScribe in conjunction with Java Script, an innovative technology is therefore provided which allows any

corresponding print system operated in accordance with the method according to the invention to be programmed just as simply as an Internet homepage. The communications possibilities already described, together with the logically modular object-oriented construction of JScribe and the Java-Script-typical expansion possibilities ideally supporting JScribe permit within the shortest possible time the construction of complex output management systems for an extremely wide range of applications.

A further preferred embodiment of the method according to the present invention is characterized in that the graphically representable objects stored in the memory in an object-oriented format, preferably also script objects (for example Java Script objects), preferably before they are output in the output print data stream, are kept ready by an application interface to be read out, to be changed, to be deleted or to have new objects appended.

According to the prior art, hitherto the page descriptions necessary for the storage of forms in the output devices had to be created laboriously by hand, that is to say programmed in the respective page description language—time-consuming and expensive work which can be carried out only by a few programmers qualified to do this. The same also applies to changes in the stored data.

The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface, by assigning the methods required for this to the respective objects in accordance with their class hierarchy. This means that the objects stored in the memory can, for example, be displayed on a screen and modified as desired. Here, too, deleting existing objects and appending new objects are also possible.

By means of binding suitable application software—also called FormMaker—it is therefore made possible in particular for each EDP user to modify existing forms and to create new forms entirely without any programming knowledge, which likewise applies to scripts.

Given suitable selection of the application interface and processing methods correspondingly available to a sufficient extent for the object classes used, a graphic core system with a functional interface is thus made available, which can be used by applications for graphic user interfaces, such as those based on the Windows operating system, to display the object data as a standard document on the screen and to modify it with different processing tools.

The application interface also preferably permits script objects, preferably Java Script objects themselves, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations being automatically transformed, if required, into script objects, preferably Java Script objects. It thus provides a complete graphic development environment for computers, preferably computers operating under the Windows operating system, which permits the printing and copying systems to be programmed without Java Script knowledge.

In addition, already existing development tools which are based on Java can likewise be used for the development of individual JScribe (registered trademark) applications.

The use of "FormMaker" application software permits the design of "intelligent" electronic forms, which are transformed into logical documents with the aid of JScribe (registered trademark). These in turn can be made available in systems connected to the network and output at any desired location by any desired printing systems, preferably laser printing systems and digital copying systems, sent as e-mail or else transferred to archiving systems.

The present method according to the invention can also be present implemented on a system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, the data processing unit being programmed in such a way that it operates in accordance with an embodiment of the method according to the invention.

In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream.

In addition, the system according to the invention can moreover permit respectively stored objects, preferably even script objects themselves, such as Java Script objects, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations being transformed automatically, if required, into Java Script objects.

The system according to the invention can also be integrated into a printer or else a printer server.

JScribe (registered trademark) can therefore not only be employed directly on printers and digital copying system but can also be implemented on PC server platforms.

For installation purposes on printing systems, the JScribe script sequences can, for example, be incorporated into a Prescribe (registered trademark) data stream. The printing system receiving this data, for example the appropriate laser printer or digital copier, will read in and compile the program code.

This permits the configuration of networks with hardware units which are small but equipped with high functionality, which have a common interface and permit access relating to archiving documents, to distributed printing (cluster printing) and security printing and much more.

The abovedescribed embodiments of the method according to the present invention can of course in each case also be implemented as a computer program product which has a computer-readable medium with computer program code means or as a computer program on an electronic carrier signal and in which, in each case after the computer program has been loaded, the computer is caused by the program to carry out the method according to the invention described here.

In the following text, an exemplary embodiment, not to be understood as restrictive, will be discussed by using the drawing, in which:

FIG. **1** shows the sequence of an embodiment of the method according to the invention using a schematic representation.

FIG. **1** shows the sequence of an embodiment of the method according to the invention using a schematic representation. From a computer **1**, an input print data stream **2** is sent to a device **3**—for example a computer such as a PC or else an intelligent output device such as an intelligent printer—which operates in accordance with the method according to the present invention. There, the input print data stream **2** is analyzed and split up by a parser **4**. The graphic objects **5**, **5***a* recognized as the product of this splitting are stored in a memory **6** in an object-oriented format; this is after they have possibly been combined to form super-objects. The objects **5** stored in the memory **6**, preferably script objects **5***a*, are kept ready to be read out via

an application interface **7**, to be changed, to be deleted or for new objects to be appended. In this way, the objects **5**, **5a** stored in the memory **6** can, for example, be displayed on a screen **9** and modified as desired. Deleting existing objects and appending new objects is also possible here. If suitable application software is used, it is thus possible for any user to modify existing forms easily and without programming knowledge or to create new forms easily and without programming knowledge or to create new forms. The graphically representable objects **5**, **5a** stored in the memory **6** in an object-oriented format are transformed into a format for the control of an output device, preferably a printer **9**, in order to be output, and the objects **5**, **5a** thus transformed are combined into an output print data stream **10** and output. Feedback messages **11** concerning the output print data stream **10** output are read in and analyzed for error messages which indicate that the printer **10** has detected a graphic object **5**, **5a** in the output print data stream **10** which cannot be output or processed by said printer. This graphic object **5**, **5a** is then split up into part objects of lower complexity, and the part objects obtained in this way, in the format for the control of the printer **9**, are slipped into the output print data stream **10** which is output to the printer **9**.

What is claimed is:

1. A method for the transformation of digital print data streams, in which

(i) an input print data stream (**2**) is read in,

(ii) this is analyzed by means of a parser (**4**) for graphically representable objects (**5**) and is split up into these graphically representable objects (**5**), and

(iii) the graphically representable objects (**5**) are stored in a memory (**6**) in an object-oriented format, and

(iv) the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format are transformed into a format for the control of an output device (**9**), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (**10**) and are output,

characterized in that graphically representable objects (**5**, **5a**) are stored in the memory (**6**) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

2. The method as claimed in claim **1**, characterized in that the graphically representable objects (**5**, **5a**) are combined into super-objects of higher complexity before being stored in the memory (**6**).

3. The method as claimed in claim **1**, characterized in that feedback messages (**11**) referring to the output print data stream (**10**) output are read in and are analyzed for error messages which indicate that the output device (**9**), preferably the printer, has recognized a transformed graphic object in the output print data stream (**10**) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (**9**), are slipped into the output print data stream (**10**) which is output to the output device (**9**).

4. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5a**) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

5. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5a**) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

6. The method as claimed in claim **5**, characterized in that the script (**5a**) automatically receiving data also requests this data automatically.

7. The method as claimed in claim **5**, characterized in that the script (**5a**) in turn reassigns the data received by it to the graphic object (**5**) associated with itself, and prints out the graphic object (**5**) assigned to itself together with the data requested, received and reassigned by itself.

8. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5a**) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

9. The method as claimed in claim **8**, characterized in that the script (**5a**) sends the graphic object (**5**) associated with itself to receiver.

10. The method as claimed in claim **9**, characterized in that the script (**5a**) in turn reassigns the data received by it to the graphic object (**5**) associated with it, and prints out the graphic object (**5**) assigned to itself together with the data requested, received and reassigned by itself.

11. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5a**) which is executed in the case of the output of the object (**5**) defined in the script (**5a**).

12. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5a**), at least one case relating to the execution of the script (**5a**) being defined in the respective script (**5a**), and occurring automatically, preferably without further influence from outside.

13. The method as claimed in claim **12**, characterized in that the automatically occurring case, defined at least in the respective script (**5a**), relating to the execution of the script (**5a**) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

14. The method as claimed in claim **13**, characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

15. The method as claimed in claim **1**, characterized in that Java Script is used as a formal language for the scripts.

16. The method as claimed in claim **1**, characterized in that the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format, preferably also script objects (**5a**), preferably before they are output in the output print data stream (**10**), are kept ready by an application interface (**7**) to be read out, to be changed, to be deleted or to have new objects (**5**) appended.

17. A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates in accordance with the method as claimed claim **1**.

18. The system for the transformation of digital print data streams as claimed in claim **17**, the system also has an operating station with display means (**8**) and input means, which makes it possible for the graphically representable objects (**5**) stored in the memory (**6**) of the data processing

unit in an object-oriented format, preferably also script objects (**5***a*), to be read out via the application interface (**7**), to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream (**10**).

19. The system for the transformation of digital print data streams as claimed in claim **17**, wherein the data processing unit, permits respectively stored objects, preferably also Java Script objects (**5***a*) themselves, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations if necessary being transformed automatically into Java Script objects (**5***a*).

20. A printer, characterized in that it has a system for the transformation of digital print data streams as claimed in claim **17**.

21. A printer server, characterized in that it has a system for the transformation of digital print data streams as claimed in claim **17**.

22. A computer-readable medium having stored thereon instructions to cause a processor to execute a method, the method comprising:

(i) an input print data stream (**2**) is read in,

(ii) this is analyzed by means of a parser (**4**) for graphically representable objects (**5**) and is split up into these graphically representable objects (**5**), and

(iii) the graphically representable objects (**5**) are stored in a memory (**6**) in an object-oriented format,

(iv) the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format are transformed into a format for the control of an output device (**9**), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (**10**) and are output,

characterized in that graphically representable objects (**5**, **5***a*) are stored in the memory (**6**) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

23. The computer-readable medium as claimed in claim **22**, the method characterized in that the graphically representable objects (**5**, **5***a*) are combined into super-objects of higher complexity before being stored in the memory (**6**).

24. The computer-readable medium as claimed in claim **22**, the method characterized in that feedback messages (**11**) referring to the output print data stream (**10**) output are read in and are analyzed for error messages which indicate that the output device (**9**), preferably the printer, has recognized a transformed graphic object in the output print data stream (**10**) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (**9**), are slipped into the output print data stream (**10**) which is output to the output device (**9**).

25. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

26. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

27. The computer-readable medium as claimed in claim **26**, the method characterized in that the script (**5***a*) automatically receiving data also requests this data automatically.

28. The computer-readable medium as claimed in claim **26**, the method characterized in that the script (**5***a*) in turn reassigns the data received by it to the graphic object (**5**) associated with itself, and prints out the graphic object (**5**) assigned to itself together with the data requested, received and reassigend by itself.

29. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

30. The computer-readable medium as claimed in claim **29**, the method characterized in that the script (**5***a*) sends the graphic object (**5**) associated with itself to a receiver.

31. The computer-readable medium as claimed in claim **30**, the method characterized in that the script (**5***a*) in turn reassigns the data received by it to the graphic object (**5**) associated with it, and forwards the graphic object (**5**) associated with itself to a receiver together with the data requested, received and reassigned by itself.

32. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which is executed in the case of the output of the object (**5**) defined in the script (**5***a*).

33. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*), at least one case relating to the execution of the script (**5***a*) being defined in the respective script (**5***a*), and occurring automatically, preferably without further influence from outside.

34. The computer-readable medium as claimed in claim **33**, the method characterized in that the automatically occurring case, defined at least in the respective script (**5***a*), relating to the execution of the script (**5***a*) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

35. The computer-readable medium as claimed in claim **34**, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

36. The computer-readable medium as claimed in claim **22**, the method characterized in that Java Script is used as a formal language for the scripts.

37. The computer-readable medium as claimed in claim **22**, the method characterized in that the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format, preferably also script objects (**5***a*), preferably before they are output in the output print data stream (**10**), are kept ready by an application interface (**7**) to be read out, to be changed, to be deleted or to have new objects (**5**) appended.

38. A computer data signal embodied in a carrier wave and representing sequences of instructions which, when executed by a processor, cause the processor to perform a method, the method comprising:

(i) an input print data stream (**2**) is read in,

(ii) this is analyzed by means of a parser (**4**) for graphically representable objects (**5**) and is split up into these graphically representable objects (**5**), and

(iii) the graphically representable objects (**5**) are stored in a memory (**6**) in an object-oriented format,

(iv) the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format are transformed into a format for the control of an output device (**9**), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (**10**) and are output,

characterized in that graphically representable objects (**5**, **5**a) are stored in the memory (**6**) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

39. The computer data signal as claimed in claim **38**, the method characterized in that the graphically representable objects (**5**, **5**a) are combined into super-objects of higher complexity before being stored in the memory (**6**).

40. The computer data signal as claimed in claim **38**, the method characterized in that feedback messages (**11**) referring to the output print data stream (**10**) output are read in and are analyzed for error messages which indicate that the output device (**9**), preferably the printer, has recognized a transformed graphic object in the output print data stream (**10**) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (**9**), are slipped into the output print data stream (**10**) which is output to the output device (**9**).

41. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5**a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

42. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5**a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

43. The computer data signal as claimed in claim **42**, the method characterized in that the script (**5**a) automatically receiving data also requests this data automatically.

44. The computer data signal as claimed in claim **42**, the method characterized in that the script (**5**a) in turn reassigns the data received by it to the graphic object (**5**) associated with itself, and prints out the graphic object (**5**) assigned to itself together with the data requested, received and reassigned by itself.

45. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5**a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

46. The computer data signal as claimed in claim **45**, the method characterized in that the script (**5**a) sends the graphic object (**5**) associated with itself to a receiver.

47. The computer data signal as claimed in claim **46**, the method characterized in that the script (**5**a) in turn reassigns the data received by it to the graphic object (**5**) associated with it, and forwards the graphic object (**5**) associated with itself to a receiver together with the data requested, received and reassigned by itself.

48. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5**a) which is executed in the case of the output of the object (**5**) defined in the script (**5**a).

49. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5**a), at least one case relating to the execution of the script (**5**a) being defined in the respective script (**5**a), and occurring automatically, preferably without further influence from outside.

50. The computer data signal as claimed in claim **49**, the method characterized in that the automatically occurring case, defined at least in the respective script (**5**a), relating to the execution of the script (**5**a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

51. The computer data signal as claimed in claim **50**, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

52. The computer data signal as claimed in claim **38**, the method characterized in that Java Script is used as a formal language for the scripts.

53. The computer data signal as claimed in claim **38**, the method characterized in that the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format, preferably also script objects (**5**a), preferably before they are output in the output print data stream (**10**), are kept ready by an application interface (**7**) to be read out, to be changed, to be deleted or to have new objects (**5**) appended.

* * * * *

Prosecution History Ser. No. 95/001,398

Date	Document
07/16/2010	REQUEST FOR INTER PARTES REEXAMINATION
08/30/2010	NOTICE OF ASSIGNMENT OF INTER PARTES REEXAMINATION REQUEST
08/30/2010	NOTICE OF INTER PARTES REEXAMINATION REQUEST FILING DATE
08/30/2010	POWER OF ATTORNEY
08/30/2010	NOTICE REGARDING CHANGE OF POWER OF ATTORNEY
10/14/2010	ORDER GRANTING INTER PARTES REEXAMINATION
11/19/2010	NON-FINAL OFFICE ACTION
12/01/2010	REQUEST FOR EXTENSION OF TIME
12/01/2010	POWER OF ATTORNEY
12/06/2010	NOTICE REGARDING CHANGE OF POWER OF ATTORNEY
12/15/2010	DECISION ON PETITION FOR EXTENSION OF TIME
01/10/2011	AMENDMENT/RESPONSE
02/02/2011	REQUEST FOR REFUND – REQUESTER
02/09/2011	THIRD PARTY REQUESTER COMMENTS
02/09/2011	INFORMATION DISCLOSURE STATEMENT
02/09/2011	PETITION UNDER 1.183 TO WAIVE PAGE LIMIT - REQUESTER
03/15/2011	NOTICE OF DEFECTIVE PAPER
03/30/2011	PETITION UNDER 1.183 TO WAIVE PAGE LIMIT - OWNER
03/30/2011	AMENDMENT/RESPONSE
04/13/2011	DECISION ON REQUESTER’S PETITION
04/26/2011	THIRD PARTY REQUESTER REVISED COMMENTS
04/26/2011	INFORMATION DISCLOSURE STATEMENT
04/26/2011	PETITION UNDER 1.183 TO WAIVE PAGE LIMIT - REQUESTER
08/23/2011	DECISION ON PETITIONS
09/06/2011	THIRD PARTY REQUESTER SECOND REVISED COMMENTS AFTER NON-FINAL ACTION
04/09/2012	ACTION CLOSING PROSECUTION
05/09/2012	RESPONSE TO ACTION CLOSING PROSECUTION
06/07/2012	THIRD PARTY REQUESTER COMMENTS
07/30/2012	RIGHT OF APPEAL NOTICE
08/27/2012	NOTICE OF APPEAL - OWNER

Prosecution History Ser. No. 95/001,398

10/29/2012	APPELLANT BRIEF - OWNER
11/28/2012	RESPONDENT BRIEF - REQUESTER
01/16/2013	EXAMINER'S ANSWER
01/29/2013	REQUEST FOR ORAL HEARING - REQUESTER
02/18/2013	REBUTTAL BRIEF - OWNER
02/18/2013	REQUEST FOR ORAL HEARING - OWNER
05/24/2013	NOTICE OF ENTRY OF REBUTTAL BRIEF
07/22/2013	PTAB DOCKETING NOTICE
07/22/2013	DOCKETING NOTICE TO APPELLANT
07/31/2013	NOTICE OF HEARING
08/06/2013	CONFIRMATION OF HEARING - REQUESTER
08/14/2013	CONFIRMATION OF HEARING - OWNER
12/18/2013	RECORD OF ORAL HEARING
12/20/2013	PTAB DECISION ON APPEAL

PTO/SB/58 (02-09)
Approved for use through 08/31/2010. OMB 0651-0033
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

(Also referred to as FORM PTO-1465)

REQUEST FOR *INTER PARTES* REEXAMINATION TRANSMITTAL FORM

Address to:
Mail Stop *Inter Partes* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Attorney Docket No.: 38512.24

Date: July 16, 2010

1. ☒ This is a request for *inter partes* reexamination pursuant to 37 CFR 1.913 of patent number 6,684,789 issued February 3, 2004. The request is made by a third party requester, identified herein below.

2. ☒ a. The name and address of the person requesting reexamination is:

   Samsung Electronics Corp, Ltd.
   Samsung Main Bldg. 250, 2-Ga, Taepyung-Ro Joong-Gu
   Seoul, Korea 100742

   b. The real party in interest (37 CFR 1.915(b)(8)) is: Samsung Electronics Corp, Ltd.

3. ☐ a. A check in the amount of $________ is enclosed to cover the reexamination fee, 37 CFR 1.20(c)(2);

   ☒ b. The Director is hereby authorized to charge the fee as set forth in 37 CFR 1.20(c)(2) to Deposit Account No. 08-1394; **or**

   ☐ c. Payment by credit card. Form PTO-2038 is attached.

4. ☒ Any refund should be made by ☐ check or ☒ credit to Deposit Account No. 08-1394
   37 CFR 1.26(c). If payment is made by credit card, refund must be to credit card account.

5. ☒ A copy of the patent to be reexamined having a double column format on one side of a separate paper is enclosed. 37 CFR 1.915(b)(5)

6. ☐ CD-ROM or CD-R in duplicate, Computer Program (Appendix) or large table
   ☐ Landscape Table on CD

7. ☐ Nucleotide and/or Amino Acid Sequence Submission
   If applicable, items a. – c. are required.

   a. ☐ Computer Readable Form (CRF)
   b. Specification Sequence Listing on:
      i. ☐ CD-ROM (2 copies) or CD-R (2 copies); **or**
      ii. ☐ paper
   c. ☐ Statements verifying identity of above copies

8. ☐ A copy of any disclaimer, certificate of correction or reexamination certificate issued in the patent is included.

9. ☒ Reexamination of claim(s) 1 - 53 is requested.

10. ☒ A copy of every patent or printed publication relied upon is submitted herewith including a listing thereof on Form PTO/SB/08, PTO-1449, or equivalent.

11. ☐ An English language translation of all necessary and pertinent non-English language patents and/or printed publications is included.

1IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent of Thomas E. Krautter	§ §	REQUEST FOR *INTER PARTES* REEXAMINATION
U.S. Patent No. 6,684,789	§ §	Attorney Docket No.: 38512.24
PCT Filed: May 11, 2001 §371(c) Date: November 7, 2002	§ § § §	Issued: February 3, 2004 Customer No.: 27683
Title: METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER	§ § § § §	Real Party in Interest: Samsung Electronics Corp, Ltd.

REQUEST FOR INTER PARTES REEXAMINATION UNDER 35 U.S.C. §§ 311-318

Mail Stop *Inter Partes* Reexam
Hon. Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Pursuant to the provisions of 35 U.S.C. §§ 311-318 (2002), Samsung Electronics Corp, Ltd. ("Samsung" or "Requester") hereby requests an *inter partes* reexamination of claims 1-53 (all of the claims) of United States Patent No. 6,684,789 ("the '789 patent," Ex. A) that issued on February 3, 2004, to Thomas E. Krautter, resulting from a PCT application filed on May 11, 2001. This request is made on behalf of Samsung, the real party in interest. Samsung certifies that the estoppel provisions of 37 C.F.R. § 1.907 do not prohibit this request for *inter partes* reexamination.

The Requester hereby asserts that claims 1-53 of the '789 patent are invalid over prior art references that were not previously before the Patent Office. Requester therefore requests that an order for reexamination be issued with an Office Action rejecting all claims.

The '789 patent is the subject of pending litigation, namely *CCP Systems AG v. Samsung Electronics Corp., Ltd., et al.*, No. 2:09-cv-04354-DMC-CCC (D. N.J. filed August 25, 2009). In the litigation, CCP Systems AG, the alleged assignee of the '789 patent and, for purposes of

I. SUMMARY OF THE PATENT AND PROSECUTION HISTORY

The '789 patent relates to a system and method for transforming digital print data streams sent to an intelligent output device such as a printer. '789 patent at 8:55-61; Fig. 1. Each of the independent claims of the '789 patent describes receiving and parsing the digital print data streams into "graphically representable objects" stored in an "object oriented format." The specification of the '789 patent describes bar-code objects and graphic objects as examples of graphically representable objects. '789 patent at 3:38-52. Each of the independent claims further describes assigning a "script" to at least one of the objects. The specification describes several examples of scripts, including a script that instructs the printer to fold the printer output in a certain way, and a script that instructs the printer to access a database interactively during the printing operation. '789 patent at 5:3-10 and 6:47-51.

During prosecution, the application for the '789 patent was allowed on the first Office action. The Examiner gave the following reasons for allowance:

REASONS FOR ALLOWANCE

1. The following is an examiner's statement of reasons for allowance:

Claims 1-53 have been indicated for allowance because the prior art fails to teach the entire combination of a method for the transformation of digital print data stream including a steps of reading an input print data stream, analyzing the input data stream by means of a parser and splitting the input data into the graphically representable objects, storing the graphically representable objects in a memory in an object-oriented format, transforming the object-oriented format into a format for controlling a printer, the object-oriented format stored in the memory including at least one stored script is assigned, which is executed in the case defined in the script.

Ex. B, Notice of Allowance mailed Sep. 30, 2003 at p. 2. For the sake of later cross-reference, the individual reasons for allowance are listed below:

a. reading an input print data stream.
b. analyzing the input data stream by means of a parser.
c. splitting the input data into the graphically representable objects
d. storing the graphically representable objects in a memory in an object-oriented format.
e. the object-oriented format stored in the memory including at least one stored script.

As will be shown in the following analysis, the prior art being presented in the present request teaches each of the above-listed reasons for allowance, and is therefore better than, and non-cumulative of, the prior art that was before the examiner during the original prosecution of the '789 patent.

II. SUBSTANTIAL NEW QUESTIONS OF PATENTABILITY

A. Citation of the Prior Art Patents and Printed Publications

In accordance with 37 C.F.R. § 1.915(b)(2), reexamination of claims 1-53 (all of the issued claims) of the '789 patent is requested in view of the following references:

Exhibit C	AS/400 Guide to Advanced Function Presentation and Print Services Facility, Fourth Edition, IBM Printing Systems, January 1999 ("IBM"), which is prior art under 35 U.S.C. § 102(b)
Exhibit D	English, Paul M. et al., Interleaf active documents, Electronic Publishing, Vol. 7(2), pp. 75-87, June 1994 ("Interleaf"), which is prior art under 35 U.S.C. § 102(b)
Exhibit E	U.S. Patent No. 5,579,519 to Pelletier, filed February 16, 1994 ("Interleaf Patent"), issued November 26, 1996, which is prior art under 35 U.S.C. 102(b)
Exhibit F	Lieberman, Henry, "Integrating user interface agents with conventional applications," Knowledge-Based Systems 11 (1998), pp. 15-23 ("Lieberman"), which is prior art under 35 U.S.C. 102(b)

B. Statements Pointing Out Each Substantial New Questions of Patentability

In this request, substantial new questions (SNQs) of patentability are raised as set forth below. For clarity, the substantial new questions of patentability are set forth in groupings based on the primary reference(s) being relied upon. The detailed explanation will address the substantial new questions of patentability in a similar order.

SNQ #1. Claims 1-5, 17, 20-26, and 38-41 are anticipated by IBM under 35 U.S.C. 102(b). An explanation as to why IBM presents a new, non-cumulative technological teaching that was not previously considered and discussed on the record during the original prosecution of the '789 patent is provided in Section C immediately below, and a detailed explanation of the pertinency and manner of applying the prior art to the claims is provided in the following section and the attached claim charts.

SNQ #2. Claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, and 48-49 are obvious over IBM in view of Interleaf under 35 U.S.C. 103(a). An explanation as to why IBM presents a new, non-cumulative technological teaching that was not previously considered and discussed on the record is provided in Section C immediately below, and a detailed explanation of the pertinency and manner of applying the prior art to the claims is provided in the following section and the attached claim charts.

SNQ #3. Claims 1-14, 16-18, 20-35, 37-51, and 53 are obvious over IBM in view of Interleaf, further in view of the Interleaf Patent under 35 U.S.C. 103(a). An explanation as to why IBM presents a new, non-cumulative technological teaching that was not previously considered and discussed on the record is provided immediately below, and a detailed explanation of the pertinency and manner of applying the prior art to the claims is provided in the following section and the attached claim charts.

SNQ #4. Claims 15, 19, 36, and 52 are obvious over IBM in view of Interleaf and further in view of the Interleaf Patent and Lieberman under 35 U.S.C. 103(a). An explanation as to why IBM presents a new, non-cumulative technological teaching that was not previously considered and discussed on the record is provided immediately below, and a detailed explanation of the pertinency and manner of applying the prior art to the claims is provided in the following section and the attached claim charts.

SNQ #5. Claims 1-2, 4-7, 11-12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 are anticipated by Interleaf under 35 U.S.C. 102(b) . An explanation as to why Interleaf presents a new, non-cumulative technological teaching that was not previously considered and discussed on the record is provided immediately below, and a detailed explanation of the pertinency and manner of applying the prior art to the claims is provided in the following section and the attached claim charts.

SNQ #6. Claims 15, 36, and 52 are obvious over Interleaf in view of Lieberman under 35 U.S.C. 103(a). An explanation as to why Interleaf presents a new, non-cumulative technological teaching that was not previously considered and discussed on the record is provided immediately below, and a detailed explanation of the pertinency and manner of applying the prior art to the claims is provided in the following section and the attached claim charts.

SNQ #7. Claims 1-2, 4-14, 15-18, 22-23, 25-35, 37-39, 41-51, and 53 are obvious over Interleaf in view of the Interleaf Patent under 35 U.S.C. 103(a). An explanation as to why Interleaf and the Interleaf Patent present a new, non-cumulative technological teaching that was not previously considered and discussed on the record during the original prosecution is provided immediately below, and a detailed explanation of the pertinency and manner of applying the prior art to the claims is provided in the following section and the attached claim charts.

SNQ #8. Claims 15, 19, 36, and 52 are obvious over Interleaf in view of the Interleaf patent, further in view of Lieberman under 35 U.S.C. 103(a). An explanation as to why Interleaf and the Interleaf patent present a new, non-cumulative technological teaching that was not previously considered and discussed on the record is provided immediately below, and a detailed explanation of the pertinency and manner of applying the prior art to the claims is provided in the following section and the attached claim charts.

SNQ #9. Claims 1-53 are obvious over the Interleaf Patent in view of IBM under 35 U.S.C. 103(a). An explanation as to why IBM and the Interleaf patent present a new, non-cumulative technological teaching that was not previously considered and discussed on the record is provided immediately below, and a detailed explanation of the pertinency and manner of applying the prior art to the claims is provided in the following section and the attached claim charts.

C. The Prior Art Of The Present Request Present New, Non-Cumulative Technological Teachings That Were Not Previously Considered And Discussed On The Record

In this request, substantial new questions (SNQs) of patentability are raised as set forth above. None of the prior art being used in the present request were of record during the original prosecution of the '789 patent, and furthermore, are new and non-cumulative of the prior art that was of record, as described below. Furthermore, the prior art of the present request is better than the prior art of record because they disclose each of the claim elements listed by the original examiner in the Reasons for Allowance.

1. <u>IBM is new and non-cumulative of the prior art of record</u>

IBM describes the Advanced Function Presentation and printing system, also called "AFP." IBM, p. 1. IBM provides detailed technical descriptions of the features of AFP that are the very same features that the Examiner believed patentable in the '789 patent. Specifically, and as listed below, IBM describes the following features of AFP that correspond exactly to the claim elements listed in the Reasons for Allowance of the '789 patent (the letters a, b, c, ... below correspond to the reasons for allowance listed on page 3, above):

a. IBM teaches reading an input print data stream. Specifically, IBM describes reading "lines of output" sent from a print application and read into the AFP. IBM, p. 193.
b. IBM teaches analyzing the input data stream by means of a parser. Specifically, IBM describes that the lines of output are "optionally parsed into individual fields." IBM. p. 194.
c. IBM teaches splitting the input data into the graphically representable objects. Specifically, the AFP's parser splits the lines of output into "print fields", which are graphically representable objects. IBM, p. 194. An example of a print field is a bar code. Id.
d. IBM teaches storing the graphically representable objects in a memory in an object-oriented format. Specifically, AFP is an "object-oriented system," with the objects being stored in "printer memory." IBM, p. 7.
e. IBM teaches the object-oriented format stored in the memory including at least one stored script. Specifically, AFP uses scripts referred to as "Data Description Specification," or DDS. IBM, p. 127. One DDS is called "ZFOLD", which is a script that instructs the printer to fold the printed paper twice, to form the shape of a "Z." IBM, pp. 134-135.

Thus, IBM teaches all of the claim elements that were listed in the Reasons for Allowance as being missing from the prior art of record during the original prosecution of the '789 patent. IBM is therefore new and non-cumulative of the prior art originally of record.

2. <u>Interleaf is new and non-cumulative of the prior art of record</u>

Interleaf describes a document processing system that has the very same features that the Examiner believed patentable in the '789 patent, discussed briefly below:

a. Interleaf teaches reading an input print data stream. Specifically, Interleaf describes reading a "document file stream." Interleaf, p. 84.

b. Interleaf teaches analyzing the input data stream by means of a parser. Specifically, Interleaf describes parsing the data stream: "If a document contains active objects within the document file stream …." Interleaf. p. 84.

c. Interleaf teaches splitting the input data into the graphically representable objects. Specifically, Interleaf describes splitting out the active objects, so that each object (referred to as the "current object") can be separately evaluated. Interleaf, p. 85.

d. Interleaf teaches storing the graphically representable objects in a memory in an object-oriented format. Specifically, Interleaf states: "The system should be object-oriented to allow for easy active document building and reusability of active objects." Interleaf, p. 82.

e. Interleaf teaches the object-oriented format stored in the memory including at least one stored script. Specifically, Interleaf describes using Lisp scripts. "Lisp scripts can be attached to any Interleaf object, stored within or external to a document." Interleaf, p. 77.

Thus, Interleaf teaches all of the claim elements that were listed in the Reasons for Allowance as being missing from the prior art of record during the original prosecution of the '789 patent. Interleaf is therefore new and non-cumulative of the prior art originally of record.

3. <u>The Interleaf Patent is new and non-cumulative of the prior art of record</u>

The Interleaf Patent describes the same document processing system as Interleaf, discussed above, and thus for the same reasons, is new and non-cumulative of the prior art of record during the original prosecution. The Interleaf patent also includes many example scripts that are the exact same scripts as listed in many of the dependent claims of the '789 patent.

An explanation of how the above-listed prior art references are better than, and non-cumulative of the cited prior art is provided below. A detailed explanation of the pertinency and manner of applying the patents and printed publications to every claim for which reexamination is requested is set forth in Section III below, with further reference to the claim charts attached as Exhibits G-L

III. DETAILED EXPLANATION OF THE PERTINENCY AND MANNER OF APPLYING THE PRIOR ART REFERENCES TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED

The following analysis is separated into three sub-sections: rejections of the claims based on IBM as the primary reference, rejections of the claims based on Interleaf as the primary reference, and rejections of the claims based on the Interleaf Patent as the primary reference.

A. Proposed Rejection of the Claims In View of IBM Alone or In Combination with Other References

As discussed in detail in the present request, IBM invalidates claims 1-53 (all of the claims) of the '789 patent either by itself, or when combined with other prior art references. An overview of IBM is provided below, followed by a description of the proposed anticipation and obviousness rejections based on IBM.

Overview of IBM

The IBM reference published on January 1999, several years before the earliest possible priority date for the '789 patent, and was not cited during the prosecution of the '789 patent. IBM is directed to presentation and printer operations of IBM's midrange computer, the AS/400. These operations are performed in a subsystem called "Advanced Function Presentation." IBM, p. 1.

> Advanced Function Printing (AFP) Data Stream is the application interface to Advanced Function Presentation (AFP). AFP Data Stream includes data and text, and device independent (not printer specific) references to AFP resources, such as overlays, page segments, font objects, and so on. AFP Data Stream is independent of operation systems and page printers, is portable across environments, and is constructed of structured fields. AFP Data Stream output is produced by a number of application enablers, such as the AFP Toolbox, Advanced Print Utility, DDS, and so on.

IBM, p 15.

AFP utilizes an object-oriented architecture for storing objects, arranged according to "structured fields." IBM, p. 21. An example of several objects for a particular AFP Data Stream is shown in Fig. 5, reproduced and annotated below.[1]

[1] It is noted that the bar code object is an example object that is described both in IBM and in the '789 patent. '789 patent at 3:46.

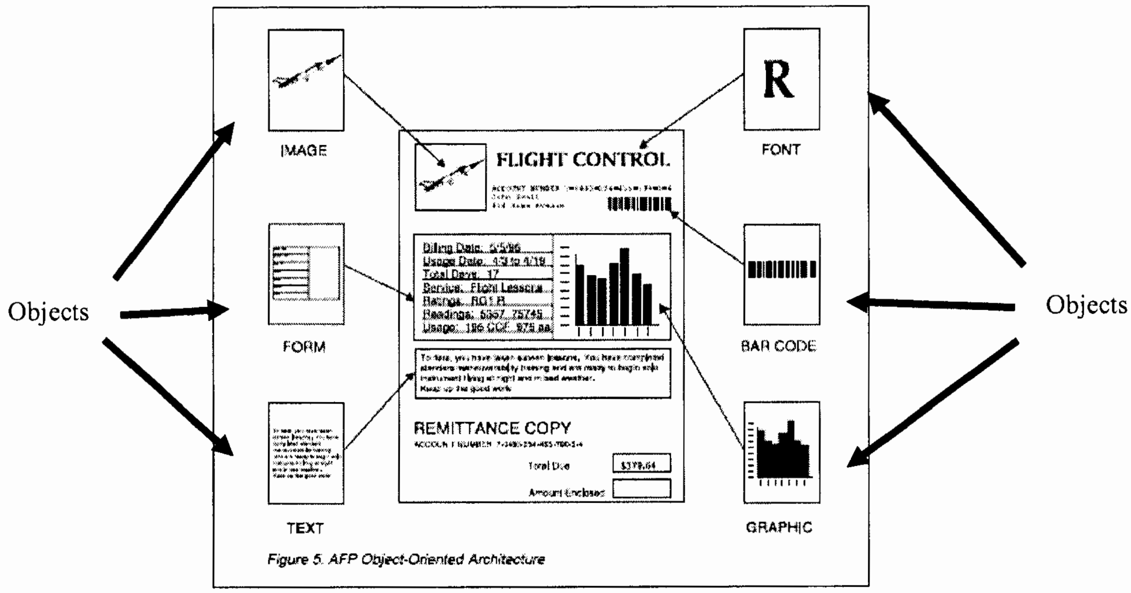


Figure 5. AFP Object-Oriented Architecture

IBM, Fig. 5, p. 23

The AFP Data Stream includes scripts referred to as "Data Description Specifications" or "DDS." IBM, p. 127. "A DDS print record is the collection of fields and/or print keywords that are to be executed when the application program issues the write command. Once the variable data is passed to DDS, DDS can control font, positioning, and other characteristics external to the application program." IBM, p. 128. One such DDS keyword is "ZFOLD", which instructs the printer to fold the paper as it is being printed.[2]

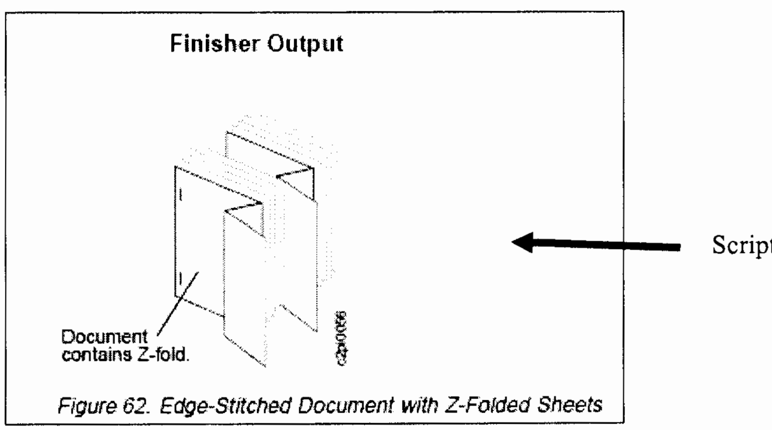


Figure 62. Edge-Stitched Document with Z-Folded Sheets

IBM, Fig. 62, p. 135

[2] It is noted that a paper folding script is an example script that is described both in IBM and in the '789 patent. '789 patent at 5:7.

1. Anticipation Rejections Based on IBM

The following is a quotation of 35 U.S.C. § 102(b) that forms the basis for all of the following anticipation rejections:

> *A person shall be entitled to a patent unless ... (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.*

Proposed Rejection #1. Claims 1-5, 17, 20-26, and 38-41 are anticipated by IBM under 35 U.S.C. § 102(b). A detailed explanation of the pertinency and manner of applying IBM to claims 1-5, 17, 20-26, and 38-41 is provided in the claim chart attached at Exhibit G.

2. Obviousness Rejections Based on IBM

The following is a quotation of 35 U.S.C. § 103(a) that forms the basis of all obviousness rejections:

> *A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.*

Proposed Rejection #2. Claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, and 48-49 are obvious over IBM in view of Interleaf under 35 U.S.C. 103(a). A detailed explanation of the pertinency and manner of applying IBM and Interleaf to claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, and 48-49 is provided in the claim chart attached at Exhibit H. The claim chart also includes reasons why one of ordinary skill in the art would combine these references.

Proposed Rejection #3. Claims 1-14, 16-18, 20-35, 37-51, and 52 are obvious over IBM in view of Interleaf, and further in view of the Interleaf Patent under 35 U.S.C. 103(a). A detailed explanation of the pertinency and manner of applying IBM, Interleaf and the Interleaf Patent to claims 1-14, 16-18, 20-35, 37-51, and 52 is provided in the claim chart attached at Exhibit I. The claim chart also includes reasons why one of ordinary skill in the art would combine these references.

Proposed Rejection #4. Claims 15, 19, 36, and 52 are obvious over IBM in view of Interleaf and further in view of the Interleaf Patent and Lieberman under 35 U.S.C. 103(a). A detailed explanation of the pertinency and manner of applying IBM, Interleaf, the Interleaf Patent, and Lieberman to claims 15, 19, 36, and 52 is provided in the claim chart attached at Exhibit I. The claim chart also includes reasons why one of ordinary skill in the art would combine these references.

B. Proposed Rejection of the Claims In View of Interleaf Alone or In Combination with Other References

As discussed in detail in the present request, the Interleaf article invalidates claims 1-53 (all of the claims) of the '789 patent either by itself, or when combined with another prior art reference. An overview of Interleaf is provided below, followed by a description of the proposed anticipation and obviousness rejections based on Interleaf.

Overview of Interleaf

The Interleaf article published on June 1994, several years before the earliest possible priority date for the '789 patent, and was not cited during the prosecution of the '789 patent. Interleaf describes a document processing system that uses "active documents", which are documents "built with an extensible object system." Interleaf, p. 75. "Document system facilities used by this application include printing, filtering to accept CALS compliant [a government standard] documents, and automatic hypertext linking and indexing." Interleaf, p. 80.

Interleaf describes receiving a "document file stream" for printing, and parses the document to see if it has any active objects:

> If a document contains active objects within the document file stream ..., these become activated when the document is opened programmatically or by the user for ... printing.

Interleaf, p. 84. An example object is a "Named Graphic Object," or "NGO."[3]

The system also evaluates each object to identify if there is an associated script. "Lisp scripts can be attached to any Interleaf object, stored within or external to a document." Interleaf, p. 77. An example script is an "auto-localizing" script, which references an external

---

[3] Interleaf, p. 81. It is noted that graphic objects are example objects that are described both in Interleaf and in the '789 patent. '789 patent at 3:42.

table to change the text to be printed according to a localized language.[4] A copy of the table is provided in Fig. 1, reproduced below.

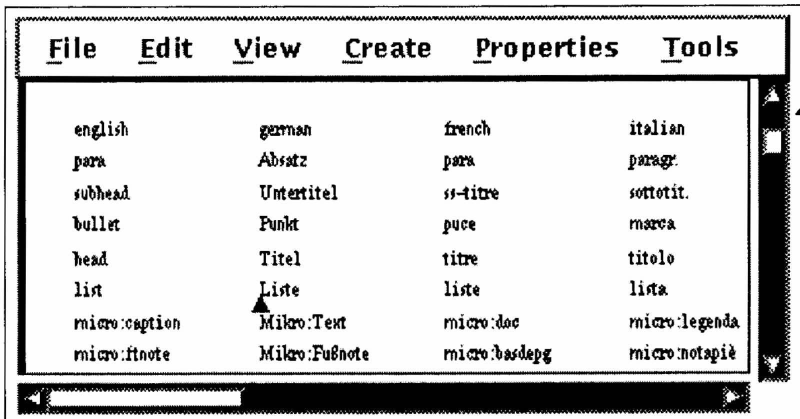


Figure 1. A template localization table

Interleaf scripts access this database of languages

Interleaf, Fig. 1, p. 78

1. <u>Anticipation Rejections Based on Interleaf</u>

The following is a quotation of 35 U.S.C. § 102(b) that forms the basis for all of the following anticipation rejections:

> *A person shall be entitled to a patent unless ... (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of the application for patent in the United States.*

<u>Proposed Rejection #5</u>. Claims 1-2, 4-7, 11-12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 are anticipated by Interleaf under 35 U.S.C. 102(b). A detailed explanation of the pertinency and manner of applying Interleaf to claims 1-2, 4-7, 11-12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 is provided in the claim chart attached at Exhibit J.

---

[4] Interleaf, p. 78. It is noted that accessing an external table/database interactively during the printing operation is an example script that is described both in Interleaf and in the '789 patent. '789 patent at 6:50-51.

C. Proposed Rejection of the Claims Over the Interleaf Patent in View Of IBM

As discussed in detail in the present request, the Interleaf Patent invalidates claims 1-53 (all of the claims) of the '789 patent either by itself, or when combined with another prior art reference. An overview of Interleaf is provided below, followed by a description of the proposed anticipation and obviousness rejections based on Interleaf.

Overview of the Interleaf Patent

The Interleaf patent has a 102(e) priority date of Feb. 16, 1994, and was not cited during the prosecution of the '789 patent. The Interleaf Patent describes an "electronic document processing system" or EDPS that modifies documents for printing. Interleaf Patent at 1:8-11; 2:10-11. Fig. 5 of the patent, reproduced below, shows an example document in the form of a memorandum 32 having multiple objects, including a "graphic object 48." Interleaf Patent, 3:4. The Interleaf Patent describes providing scripts to the objects to trigger events during presentation or printing. A list of example events is provided in columns 6-9 of the patent.

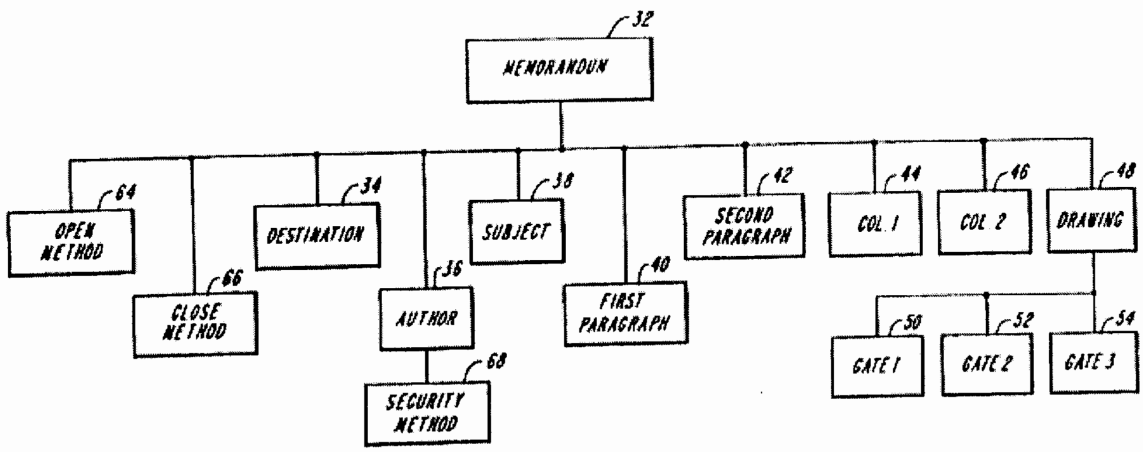


FIG. 5

The Interleaf Patent, FIG. 5

1. <u>Obviousness Rejections Based on Interleaf</u>

The following is a quotation of 35 U.S.C. § 103(a) that forms the basis of all obviousness rejections:

> *A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the*

US006684789B2

(12) **United States Patent**
Krautter

(10) **Patent No.: US 6,684,789 B2**
(45) **Date of Patent: Feb. 3, 2004**

(54) **METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER**

(75) Inventor: **Thomas Erfinders Krautter**, Stuttgart (DE)

(73) Assignee: **CCP Systems AG**, Stuttgart (DE)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **10/275,784**

(22) PCT Filed: **May 11, 2001**

(86) PCT No.: **PCT/DE01/01796**

§ 371 (c)(1),
(2), (4) Date: **Nov. 7, 2002**

(87) PCT Pub. No.: **WO01/88840**

PCT Pub. Date: **Nov. 22, 2001**

(65) **Prior Publication Data**

US 2003/0140809 A1 Jul. 31, 2003

(30) **Foreign Application Priority Data**

May 17, 2000 (DE) 100 24 177
Jan. 26, 2001 (DE) 101 03 733

(51) **Int. Cl.**[7] **B41F 1/54**
(52) **U.S. Cl.** **101/484**; 101/486; 400/61; 400/62
(58) **Field of Search** 101/484, 485, 101/486; 400/61, 62, 63, 76; 358/1.1, 1.9, 1.15, 1.16, 1.17, 1.18

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,216,754 A		6/1993	Sathi et al.	
5,566,278 A	*	10/1996	Patel et al.	358/1.15
6,006,013 A		12/1999	Rumph et al.	

FOREIGN PATENT DOCUMENTS

EP	0 109 615 B1	6/1994
EP	0 964 339 A2	12/1999
EP	1 061 456 A2	12/2000
GB	2 357 348 A	6/2001
WO	WO-00/17748	3/2000
WO	WO-00/55720	9/2000

* cited by examiner

Primary Examiner—Andrew H. Hirshfeld
Assistant Examiner—Minh Chau
(74) *Attorney, Agent, or Firm*—Squire, Sanders & Dempsey L.L.P.

(57) **ABSTRACT**

A method for the transformation of digital print data streams, in which an input print data stream (**2**) is read in, this is analyzed by means of a parser (**4**) for graphically representable objects (**5**, **5***a*) and is split up into these graphically representable objects (**5**, **5***a*), and the graphically representable objects (**5**, **5***a*) are stored in a memory (**6**) in an object-oriented format, and the graphically representable objects (**5**, **5***a*) stored in the memory (**6**) in an object-oriented format are transformed into a format for the control of an output device (**9**), preferably a printer, and the objects thus transformed are combined into an output print data stream (**10**) and are output, graphically representable objects (**5**, **5***a*) being stored in the memory (**6**) in an object-oriented format, to which at least one stored script (**5***a*) is assigned, which is executed in the cases defined in the script (**5***a*).

53 Claims, 1 Drawing Sheet

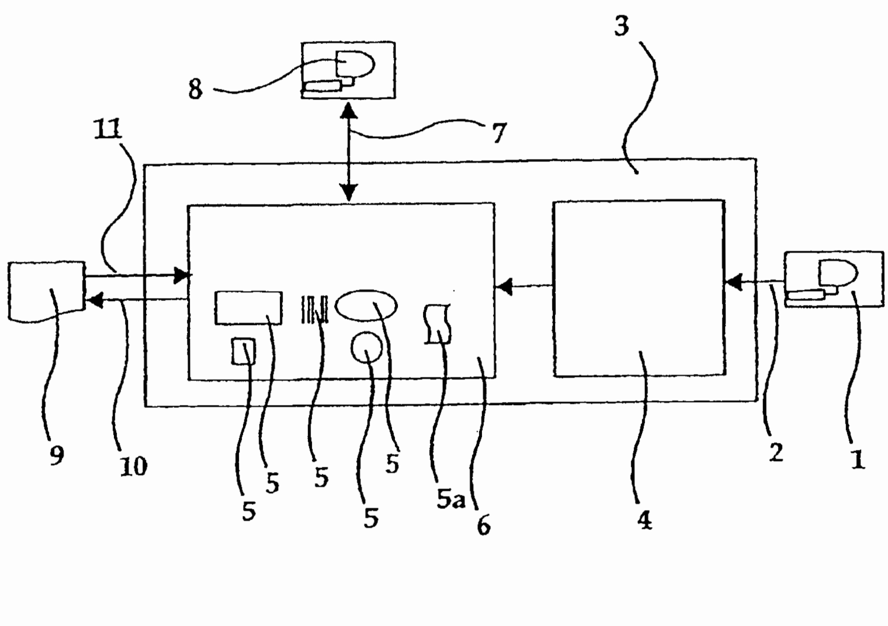
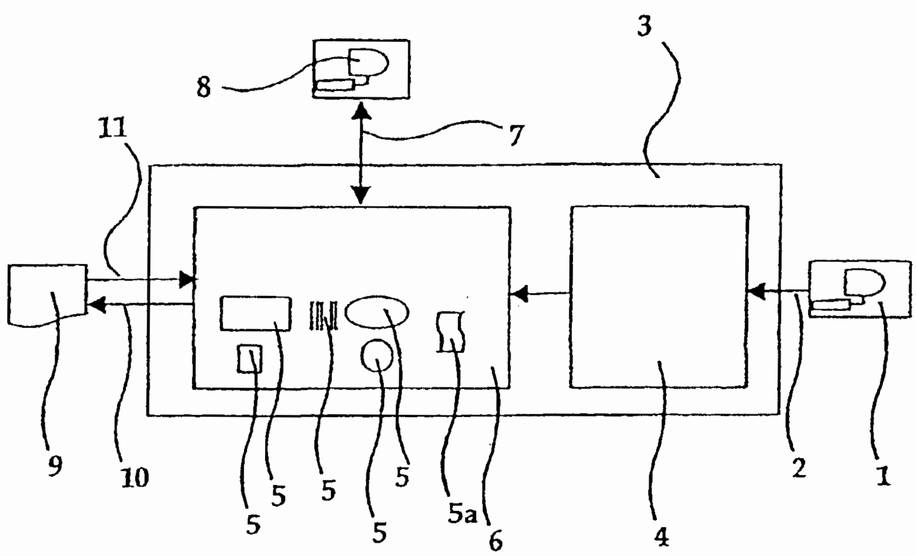


FIG. 1



METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER

The present invention relates to a method and system for the transformation of digital print data streams and corresponding printer and printer server.

Virtually all the output devices which are common nowadays use "page description languages", also called PDL, to produce printed documents. Here, an application program controls a driver for the output device (for example a printer driver). This driver converts information about the graphic objects to be output—for example text or image information—into the respective PDL suitable for the printer used, so that the latter can hereby be controlled directly.

More recent output devices, such as laser printers or digital color printers for example, also offer the possibility of buffering the data streams coming in to control them and, for example, using them as an original form for further incoming print data. This makes it possible to dispense with forms needed for the respective printing, such as letter paper, invoice forms or the like for example, in the individual case. Instead, the application software respectively used merely calls up the form stored once in the printer and combines it with the current print data. In this way, the accumulation of data, for example in networks, can be reduced considerably. However, the result is also organizational advantages: since the forms used no longer have to be kept in reserve by each individual user on his computer, in this way standardized use forms can be achieved, which firstly helps to ensure the often desired standard appearance of a company or an institution and secondly also makes it easier to use current form versions.

However, these aforementioned advantages are normally not used, since the printers used in a company or an institution—with regard to their control—are often not uniform and therefore the use of the functions described above is too complicated, since the appropriate forms either have to be available for each printer model used, which would be very labor-intensive, or only specific printers can be used for specific applications, which is very inflexible.

One possibility of solving this problem is to circumvent the abovedescribed inhomogeneity of the output devices used by employing methods for the conversion of various data stream formats for controlling output devices, which makes it possible for all the computers which produce print data streams to be output to use a standard format for this purpose, by each printer being assigned an interface—be it a dedicated device, be it merely in the form of a software filter—which makes use of such a method and, on the side of the input data stream, uses the format to be used uniformly and, on the side of the output data stream, uses the specific format of the printer to be controlled.

Such a solution is described, for example, by EP 0 109 615 B1, which refers to a method for the conversion of text which is represented in the form of digital data. However, the method taught by this document has considerable disadvantages with regard to the possibilities of current systems from information technology: for example, the method is suitable only for those input print data streams which, in their syntax, follow a format description language whose syntax may be described with the aid of "regular expressions". This is because the method taught in EP 0 109 615 B1 makes use of a status machine, implemented by means of "key status variables", for the recognition and conversion of input control objects recognized in the input print data

stream into output control objects. These output control objects are in this case produced directly from the input control objects—specifically in accordance with a fixed assignment—as a function of the respective state representing the key status variables. Such a procedure corresponds to the functioning of the theoretical model of the Moore or Mealy machines, which operate quite efficiently but permit only, the recognition of regular expressions. For these circumstances surrounding information technology at the priority date of EP 0 109 615, such a simple possible transformation may have been sufficient, since—as can already be gathered from claim 1 there—only text had to be converted, apart from format information.

For the current circumstances of PDLs or else other input formats to be recognized where possible, such as HTML or XML, this no longer applies in any way, however. In the meantime, these have been built up in such a complex way with regard to their possibilities that a status machine is no longer in any way adequate for their recognition and conversion.

However, the target format, into which the print data stream is to be transformed, nowadays places high requirements on a transformation: although in principle there would be the possibility here likewise of using the smallest common multiple of the functions of current printing format and in this way of reducing the effort on transformation, this convenience in the design of the transformation process would be brought at great expense in the operation of the method, since in this way the accumulation of data in networks would be increased again, since powerful printer control possibilities which as a rule become more and more specific with regard to the printer type used as the complexity increases, would necessarily have to be dispensed with. Such an increased accumulation of data would, however, again stand in the way of the objective of reducing the data traffic in the network by using PDLs. Thus, at the same time, there is a requirement on the transformation process that the latter produces the preconditions that the target formats can be produced in the most flexible manner possible with all their available printing functions, in order that the traffic on the data transmission lines can thus be minimized.

Furthermore, it is necessary to state that printing systems, even today, still only fulfill a single purpose: namely printing. All the manufacturers of laser printers and digital copying systems have made great efforts in recent years to match the processor powers, storage capacities and additional options (such as memory cards, hard drives, network cards) of these systems to the increasing requirements. However, the manner in which printers and copiers are controlled and programmed has not changed significantly in the last ten years.

Printing systems are still controlled by a page description language (PDL) such as PCL, Postscript or Prescribe. It permits a document and its components to be described adequately. However, the many additional options of modem printing and copying systems available in the meantime cannot be used. The consequence of this is that, even today, the entire printing process is controlled and monitored by a host computer. Its task substantially comprises converting the respective information exactly into the page description language "understood" by the printing system.

It is therefore an object of the present invention to specify a method for the transformation of digital print data streams which is both capable of recognizing more complex page description languages, whose syntax may no longer be described with the aid of simple regular expressions, and also provides the preconditions that the recognized graphic

objects can be transformed into a target format, but also processed further, as flexibly and effectively as possible, that is to say with regard to their description at the highest possible level of abstraction.

According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in, this is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects, and the graphically representable objects are stored in a memory in an object-oriented format, and the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer, and the objects thus transformed are combined into an output print data stream and are output, and which, according to the invention, is characterized in that graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

In this case, as opposed to the use of a status machine, the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used. Instead, such a parser, in terms of its theoretical performance, corresponds to a Turing machine and therefore ensures the theoretically maximum achievable performance for the analysis and splitting up of formal languages.

Furthermore, storing the graphically representable objects—and therefore of course also the scripts, which themselves are certainly also graphically representable objects—in a memory in an object-oriented format achieves the situation where the objects recognized by the parser are then available in this intermediate format which is extremely beneficial for further processing.

The objects are preferably managed here by means of a "display list management", which supports one-page and multi-page documents at as many levels as desired and which can be expanded dynamically by new objects. The individual graphic objects are stored by using their membership of specific—expediently suitably hierarchically organized—classes such as relating to the class of points, ellipses, circles, lines, polygons, rectangles, squares or else to the more complex object types, such as bar codes, more complex texts or freely definable elements such as color profiles or fonts, which permits their effective conversion into an output print data stream, since, through the class of the respective object, there is already implicit information available about its possible transformation into the format of the output print data stream. For example, via an object of the type square, it is already known from the object hierarchy that this is a subclass of the rectangle. If, then, the target format for which an output print data stream is to be produced provides speech constructs relating to the description of rectangles in the page description language, then it is clear, merely on the basis of the position of the square in the object class hierarchy, that this is also a rectangle—albeit with special characteristics—and to this extent the possibilities of the target format with regard to rectangles can also be used for an object in the square class.

In addition to such implicit information—which can be derived from the object class hierarchy—about the individual objects, however, it is also possible to add to the objects explicit information about their possible conversions into specific target formats, it being advantageously possible for this also to be combined with the abovedescribed implicitly provided information, for example by a conversion method into a specific target format being added to a class which is arranged higher in the object class hierarchy, and then automatically also being available to the objects of subordinate, lower-ranking classes by way of inheritance, if a better specified method is not already assigned to said subordinate classes.

In one embodiment of the method according to the invention, the graphically representable objects are combined into super-objects of higher complexity before being stored in the memory.

The super-objects obtained in this way are then stored in the memory in the object-oriented format. In this way, less complex graphic objects can be combined to form more complex graphic super-objects. For example, sequences of lines which in each case join one another at the ends and have been recognized as graphic objects in the input print data stream can be combined to form a graphic polygon super-object. Such a combination offers various advantages, such as easier handling of the super-object stored as a whole as compared with the individual objects, since said super-object can then be treated uniformly by the methods for the super-object class with effect for all the part objects combined in it. It also helps, in certain circumstances, to further minimize the data traffic on the transmission lines used, since an object once combined is subsequently also forwarded in combined form in the output print data stream—if technically supported there—which generally requires a lower data volume to be transmitted than the transmission of the individual objects.

A preferred embodiment of the method according to the present invention is characterized in that a parser is used for the analysis and splitting up into the graphically representable objects, which, in the theoretical model, corresponds to an automatic push-down facility and which is therefore capable of analyzing and splitting up languages with "context-free grammars" particularly effectively.

A further embodiment of the method according to the present invention is characterized in that feedback messages referring to the output print data stream output are read in and are analyzed for error messages which indicate that the output device, preferably the printer, has recognized a transformed graphic object in the output print data stream which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device, are slipped into the output print data stream which is output to the output device.

In this way, it is likewise possible to test whether the driven output device is, for example, capable of recognizing and outputting a bar-code object directly or not. If it is not capable of this and reports this back, then the bar code is simply split up into objects of the next lower hierarchy, for example filled rectangles, and a further try is made with these objects. This is continued until—if necessary until the graphic objects are split up into individual points—the output attempt is successful. The object-oriented data structure with its object hierarchy, chosen for the intermediate format, also proves to be particularly suitable for this procedure. For the further performance of the method, it is preferably noted at which level of the object classes in each case the splitting process was successful for a specific output device, in order then, in the next attempt, already to begin the output process at this level, in order also thus to avoid unnecessary data transfers, but likewise to utilize the maximum level of abstraction of the output device. In this way,

the data volume to be transmitted is reduced to the necessary extent, even with high flexibility.

In an embodiment of the method according to the present invention, at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment, which permits the incorporation of all the devices needed in the widest sense for document processing.

A further preferred embodiment of the method according to the invention is characterized in that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

The script automatically receiving data can preferably also request this data automatically.

It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver.

It can also in turn reassign the data received by it to the graphic object associated with it, and forwards the graphic object associated with itself to a receiver together with the data requested, received and reassigned by itself, or else print out said data.

In relation to the above explanations, it should be noted that the embodiments of the method according to the invention which themselves provide other objects with objects, for example by forwarding them or keeping them ready to receive or for interrogation by a script, for example, are also covered by the term "dynamic object linking" (DOL).

Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails and of printing original print and image data without a printer driver. They are able to store any desired information on hard disks or memory cards and make said data available to all the devices connected in the network and Internet. In other words, they independently undertake demanding tasks in information processing and provision, in order to relieve host computers and personal computers of quite a lot of administrative tasks. In a heterogeneous network and printing environment with laser printing and copying systems from different manufacturers in combination with impact printers and special printing systems, they also make it possible to administer all the connected printing systems with the aid of a single standardized programming language, namely the script language, and therefore reduce the effort on administration to a minimum. At this point, it should be mentioned that these systems operating by the method according to the invention are also designated JScribe (registered trademark) systems and, accordingly, the method according to the invention is also designated JScribe (registered trademark).

When JScribe (registered trademark) is used, developers and system houses will therefore be in a position to provide objects and functions which are stored in resident form in the printing system and permit and control desired individual operating sequences. These objects and functions can use any functionality provided by the JScribe (registered trademark) basic technology, including extremely demanding commands for the job or page processing and for the complete control of the print data and emulations. The method according to the invention preferably also enables access to internal printer functions and status information (page counter, network components, file system and so on), for example via a script.

The method according to the invention is preferably characterized in that graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the case of the output of the object defined in the script. In this way, for example, it is possible to execute such scripts, for example Visual Basic Scripts, Java Scripts or else "stream code" in an event-oriented manner, for example in the case where a form object is printed out, likewise "ON-PRINT" by which means, for example, to execute such functions as the printing of copies of the same form with the same net data but on different paper from different trays. In particular in interaction with those embodiments of the method according to the invention which control external devices, such as folding or enveloping machines or else stapling machines, this is particularly advantageous.

However, it may also be the case that at least one case relating to the execution of the script is defined in the respective script and occurs automatically, preferably without further influence from outside.

For example, the automatically occurring case, defined at least in the respective script, relating to the execution of the script can be configured as a timer, that is to say as a case which occurs automatically as a result of the expiry of a time, this timer preferably operating cyclically, that is to say starting itself again upon expiry.

Automatic scripts can therefore intrinsically become active and, for example, load the daily newspaper, where possible itself assembled from different sources, from the Internet, assign the found, loaded and analyzed information to a stored object and then print this object, completely without the participation of a PC or other host computer to which the printer would be connected.

For example, the simple download of JScribe (registered trademark) sequences (scripts with appropriately associated objects) can, for example, arrange for the printer automatically to fetch information about current share prices, to format it and to print it out. Image information, text documents, web pages, XML documents and any other desired print data can be analyzed while dispensing with any preparation by the PC (for example by a printer driver), modified if necessary and printed out in optimum quality. Since JScribe (registered trademark) can also be employed simultaneously as a server version for computer systems, printing systems are for the first time made capable of accessing stocks of data on host systems (for example SQL databases) interactively during the printing operation.

The language used for the scripts according to the present invention is preferably Java Script. Java Script, as a world-established standard for the script-controlled, intelligent programming of web pages, has triggered in the Internet an avalanche of innovative and functional solutions which have contributed decisively to ringing in the age of eBusiness and eCommerce. This intelligent technology, which has so decisively marked the worldwide, rapid development of the Internet, is therefore now also available for printing systems for the first time and here preferably forms the basic technology for script applications in the area of the present invention, and consequently print and document management, which is certainly uniquely and, as compared with established solutions, considerably more cost-effective.

With JScribe in conjunction with Java Script, an innovative technology is therefore provided which allows any

corresponding print system operated in accordance with the method according to the invention to be programmed just as simply as an Internet homepage. The communications possibilities already described, together with the logically modular object-oriented construction of JScribe and the Java-Script-typical expansion possibilities ideally supporting JScribe permit within the shortest possible time the construction of complex output management systems for an extremely wide range of applications.

A further preferred embodiment of the method according to the present invention is characterized in that the graphically representable objects stored in the memory in an object-oriented format, preferably also script objects (for example Java Script objects), preferably before they are output in the output print data stream, are kept ready by an application interface to be read out, to be changed, to be deleted or to have new objects appended.

According to the prior art, hitherto the page descriptions necessary for the storage of forms in the output devices had to be created laboriously by hand, that is to say programmed in the respective page description language—time-consuming and expensive work which can be carried out only by a few programmers qualified to do this. The same also applies to changes in the stored data.

The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface, by assigning the methods required for this to the respective objects in accordance with their class hierarchy. This means that the objects stored in the memory can, for example, be displayed on a screen and modified as desired. Here, too, deleting existing objects and appending new objects are also possible.

By means of binding suitable application software—also called FormMaker—it is therefore made possible in particular for each EDP user to modify existing forms and to create new forms entirely without any programming knowledge, which likewise applies to scripts.

Given suitable selection of the application interface and processing methods correspondingly available to a sufficient extent for the object classes used, a graphic core system with a functional interface is thus made available, which can be used by applications for graphic user interfaces, such as those based on the Windows operating system, to display the object data as a standard document on the screen and to modify it with different processing tools.

The application interface also preferably permits script objects, preferably Java Script objects themselves, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations being automatically transformed, if required, into script objects, preferably Java Script objects. It thus provides a complete graphic development environment for computers, preferably computers operating under the Windows operating system, which permits the printing and copying systems to be programmed without Java Script knowledge.

In addition, already existing development tools which are based on Java can likewise be used for the development of individual JScribe (registered trademark) applications.

The use of "FormMaker" application software permits the design of "intelligent" electronic forms, which are transformed into logical documents with the aid of JScribe (registered trademark). These in turn can be made available in systems connected to the network and output at any desired location by any desired printing systems, preferably laser printing systems and digital copying systems, sent as e-mail or else transferred to archiving systems.

The present method according to the invention can also be present implemented on a system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, the data processing unit being programmed in such a way that it operates in accordance with an embodiment of the method according to the invention.

In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream.

In addition, the system according to the invention can moreover permit respectively stored objects, preferably even script objects themselves, such as Java Script objects, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations being transformed automatically, if required, into Java Script objects.

The system according to the invention can also be integrated into a printer or else a printer server.

JScribe (registered trademark) can therefore not only be employed directly on printers and digital copying system but can also be implemented on PC server platforms.

For installation purposes on printing systems, the JScribe script sequences can, for example, be incorporated into a Prescribe (registered trademark) data stream. The printing system receiving this data, for example the appropriate laser printer or digital copier, will read in and compile the program code.

This permits the configuration of networks with hardware units which are small but equipped with high functionality, which have a common interface and permit access relating to archiving documents, to distributed printing (cluster printing) and security printing and much more.

The abovedescribed embodiments of the method according to the present invention can of course in each case also be implemented as a computer program product which has a computer-readable medium with computer program code means or as a computer program on an electronic carrier signal and in which, in each case after the computer program has been loaded, the computer is caused by the program to carry out the method according to the invention described here.

In the following text, an exemplary embodiment, not to be understood as restrictive, will be discussed by using the drawing, in which:

FIG. **1** shows the sequence of an embodiment of the method according to the invention using a schematic representation.

FIG. **1** shows the sequence of an embodiment of the method according to the invention using a schematic representation. From a computer **1**, an input print data stream **2** is sent to a device **3**—for example a computer such as a PC or else an intelligent output device such as an intelligent printer—which operates in accordance with the method according to the present invention. There, the input print data stream **2** is analyzed and split up by a parser **4**. The graphic objects **5**, **5***a* recognized as the product of this splitting are stored in a memory **6** in an object-oriented format; this is after they have possibly been combined to form super-objects. The objects **5** stored in the memory **6**, preferably script objects **5***a*, are kept ready to be read out via

an application interface **7**, to be changed, to be deleted or for new objects to be appended. In this way, the objects **5**, **5***a* stored in the memory **6** can, for example, be displayed on a screen **9** and modified as desired. Deleting existing objects and appending new objects is also possible here. If suitable application software is used, it is thus possible for any user to modify existing forms easily and without programming knowledge or to create new forms easily and without programming knowledge or to create new forms. The graphically representable objects **5**, **5***a* stored in the memory **6** in an object-oriented format are transformed into a format for the control of an output device, preferably a printer **9**, in order to be output, and the objects **5**, **5***a* thus transformed are combined into an output print data stream **10** and output. Feedback messages **11** concerning the output print data stream **10** output are read in and analyzed for error messages which indicate that the printer **10** has detected a graphic object **5**, **5***a* in the output print data stream **10** which cannot be output or processed by said printer. This graphic object **5**, **5***a* is then split up into part objects of lower complexity, and the part objects obtained in this way, in the format for the control of the printer **9**, are slipped into the output print data stream **10** which is output to the printer **9**.

What is claimed is:

1. A method for the transformation of digital print data streams, in which

(i) an input print data stream (**2**) is read in,

(ii) this is analyzed by means of a parser (**4**) for graphically representable objects (**5**) and is split up into these graphically representable objects (**5**), and

(iii) the graphically representable objects (**5**) are stored in a memory (**6**) in an object-oriented format, and

(iv) the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format are transformed into a format for the control of an output device (**9**), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (**10**) and are output,

characterized in that graphically representable objects (**5**, **5***a*) are stored in the memory (**6**) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

2. The method as claimed in claim **1**, characterized in that the graphically representable objects (**5**, **5***a*) are combined into super-objects of higher complexity before being stored in the memory (**6**).

3. The method as claimed in claim **1**, characterized in that feedback messages (**11**) referring to the output print data stream (**10**) output are read in and are analyzed for error messages which indicate that the output device (**9**), preferably the printer, has recognized a transformed graphic object in the output print data stream (**10**) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (**9**), are slipped into the output print data stream (**10**) which is output to the output device (**9**).

4. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

5. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

6. The method as claimed in claim **5**, characterized in that the script (**5***a*) automatically receiving data also requests this data automatically.

7. The method as claimed in claim **5**, characterized in that the script (**5***a*) in turn reassigns the data received by it to the graphic object (**5**) associated with itself, and prints out the graphic object (**5**) assigned to itself together with the data requested, received and reassigned by itself.

8. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

9. The method as claimed in claim **8**, characterized in that the script (**5***a*) sends the graphic object (**5**) associated with itself to receiver.

10. The method as claimed in claim **9**, characterized in that the script (**5***a*) in turn reassigns the data received by it to the graphic object (**5**) associated with it, and prints out the graphic object (**5**) assigned to itself together with the data requested, received and reassigned by itself.

11. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which is executed in the case of the output of the object (**5**) defined in the script (**5***a*).

12. The method as claimed in claim **1**, characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*), at least one case relating to the execution of the script (**5***a*) being defined in the respective script (**5***a*), and occurring automatically, preferably without further influence from outside.

13. The method as claimed in claim **12**, characterized in that the automatically occurring case, defined at least in the respective script (**5***a*), relating to the execution of the script (**5***a*) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

14. The method as claimed in claim **13**, characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

15. The method as claimed in claim **1**, characterized in that Java Script is used as a formal language for the scripts.

16. The method as claimed in claim **1**, characterized in that the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format, preferably also script objects (**5***a*), preferably before they are output in the output print data stream (**10**), are kept ready by an application interface (**7**) to be read out, to be changed, to be deleted or to have new objects (**5**) appended.

17. A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates in accordance with the method as claimed claim **1**.

18. The system for the transformation of digital print data streams as claimed in claim **17**, the system also has an operating station with display means (**8**) and input means, which makes it possible for the graphically representable objects (**5**) stored in the memory (**6**) of the data processing

unit in an object-oriented format, preferably also script objects (**5***a*), to be read out via the application interface (**7**), to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream (**10**).

19. The system for the transformation of digital print data streams as claimed in claim **17**, wherein the data processing unit, permits respectively stored objects, preferably also Java Script objects (**5***a*) themselves, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations if necessary being transformed automatically into Java Script objects (**5***a*).

20. A printer, characterized in that it has a system for the transformation of digital print data streams as claimed in claim **17**.

21. A printer server, characterized in that it has a system for the transformation of digital print data streams as claimed in claim **17**.

22. A computer-readable medium having stored thereon instructions to cause a processor to execute a method, the method comprising:

(i) an input print data stream (**2**) is read in,

(ii) this is analyzed by means of a parser (**4**) for graphically representable objects (**5**) and is split up into these graphically representable objects (**5**), and

(iii) the graphically representable objects (**5**) are stored in a memory (**6**) in an object-oriented format,

(iv) the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format are transformed into a format for the control of an output device (**9**), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (**10**) and are output,

characterized in that graphically representable objects (**5**, **5***a*) are stored in the memory (**6**) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

23. The computer-readable medium as claimed in claim **22**, the method characterized in that the graphically representable objects (**5**, **5***a*) are combined into super-objects of higher complexity before being stored in the memory (**6**).

24. The computer-readable medium as claimed in claim **22**, the method characterized in that feedback messages (**11**) referring to the output print data stream (**10**) output are read in and are analyzed for error messages which indicate that the output device (**9**), preferably the printer, has recognized a transformed graphic object in the output print data stream (**10**) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (**9**), are slipped into the output print data stream (**10**) which is output to the output device (**9**).

25. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

26. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

27. The computer-readable medium as claimed in claim **26**, the method characterized in that the script (**5***a*) automatically receiving data also requests this data automatically.

28. The computer-readable medium as claimed in claim **26**, the method characterized in that the script (**5***a*) in turn reassigns the data received by it to the graphic object (**5**) associated with itself, and prints out the graphic object (**5**) assigned to itself together with the data requested, received and reassigend by itself.

29. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

30. The computer-readable medium as claimed in claim **29**, the method characterized in that the script (**5***a*) sends the graphic object (**5**) associated with itself to a receiver.

31. The computer-readable medium as claimed in claim **30**, the method characterized in that the script (**5***a*) in turn reassigns the data received by it to the graphic object (**5**) associated with it, and forwards the graphic object (**5**) associated with itself to a receiver together with the data requested, received and reassigned by itself.

32. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which is executed in the case of the output of the object (**5**) defined in the script (**5***a*).

33. The computer-readable medium as claimed in claim **22**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*), at least one case relating to the execution of the script (**5***a*) being defined in the respective script (**5***a*), and occurring automatically, preferably without further influence from outside.

34. The computer-readable medium as claimed in claim **33**, the method characterized in that the automatically occurring case, defined at least in the respective script (**5***a*), relating to the execution of the script (**5***a*) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

35. The computer-readable medium as claimed in claim **34**, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

36. The computer-readable medium as claimed in claim **22**, the method characterized in that Java Script is used as a formal language for the scripts.

37. The computer-readable medium as claimed in claim **22**, the method characterized in that the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format, preferably also script objects (**5***a*), preferably before they are output in the output print data stream (**10**), are kept ready by an application interface (**7**) to be read out, to be changed, to be deleted or to have new objects (**5**) appended.

38. A computer data signal embodied in a carrier wave and representing sequences of instructions which, when executed by a processor, cause the processor to perform a method, the method comprising:

(i) an input print data stream (**2**) is read in,

(ii) this is analyzed by means of a parser (**4**) for graphically representable objects (**5**) and is split up into these graphically representable objects (**5**), and

(iii) the graphically representable objects (**5**) are stored in a memory (**6**) in an object-oriented format,

(iv) the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format are transformed into a format for the control of an output device (**9**), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (**10**) and are output,

characterized in that graphically representable objects (**5**, **5***a*) are stored in the memory (**6**) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

39. The computer data signal as claimed in claim **38**, the method characterized in that the graphically representable objects (**5**, **5***a*) are combined into super-objects of higher complexity before being stored in the memory (**6**).

40. The computer data signal as claimed in claim **38**, the method characterized in that feedback messages (**11**) referring to the output print data stream (**10**) output are read in and are analyzed for error messages which indicate that the output device (**9**), preferably the printer, has recognized a transformed graphic object in the output print data stream (**10**) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (**9**), are slipped into the output print data stream (**10**) which is output to the output device (**9**).

41. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

42. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

43. The computer data signal as claimed in claim **42**, the method characterized in that the script (**5***a*) automatically receiving data also requests this data automatically.

44. The computer data signal as claimed in claim **42**, the method characterized in that the script (**5***a*) in turn reassigns the data received by it to the graphic object (**5**) associated with itself, and prints out the graphic object (**5**) assigned to itself together with the data requested, received and reassigned by itself.

45. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

46. The computer data signal as claimed in claim **45**, the method characterized in that the script (**5***a*) sends the graphic object (**5**) associated with itself to a receiver.

47. The computer data signal as claimed in claim **46**, the method characterized in that the script (**5***a*) in turn reassigns the data received by it to the graphic object (**5**) associated with it, and forwards the graphic object (**5**) associated with itself to a receiver together with the data requested, received and reassigned by itself.

48. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*) which is executed in the case of the output of the object (**5**) defined in the script (**5***a*).

49. The computer data signal as claimed in claim **38**, the method characterized in that at least one graphically representable object (**5**) stored in the memory (**6**) in the object-oriented format is assigned at least one script (**5***a*), at least one case relating to the execution of the script (**5***a*) being defined in the respective script (**5***a*), and occurring automatically, preferably without further influence from outside.

50. The computer data signal as claimed in claim **49**, the method characterized in that the automatically occurring case, defined at least in the respective script (**5***a*), relating to the execution of the script (**5***a*) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

51. The computer data signal as claimed in claim **50**, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

52. The computer data signal as claimed in claim **38**, the method characterized in that Java Script is used as a formal language for the scripts.

53. The computer data signal as claimed in claim **38**, the method characterized in that the graphically representable objects (**5**) stored in the memory (**6**) in an object-oriented format, preferably also script objects (**5***a*), preferably before they are output in the output print data stream (**10**), are kept ready by an application interface (**7**) to be read out, to be changed, to be deleted or to have new objects (**5**) appended.

* * * * *

REASONS FOR ALLOWANCE

1. The following is an examiner's statement of reasons for allowance:

Claims 1-53 have been indicated for allowance because the prior art fails to teach the entire combination of a method for the transformation of digital print data stream including a steps of reading an input print data stream, analyzing the input data stream by means of a parser and splitting the input data into the graphically representable objects, storing the graphically representable objects in a memory in an object-oriented format, transforming the object-oriented format into a format for controlling a printer, the object-oriented format stored in the memory including at least one stored script is assigned, which is executed in the case defined in the script.

2. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

3. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Minh H Chau whose telephone number is (703) 305-0298. The examiner can normally be reached on M - TH.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Andrew H Hirshfeld can be reached on (703) 305-6619. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 308-0956.

MHC
September 30, 2003

ANDREW H. HIRSHFELD
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2800

AS/400

AS/400 Guide to Advanced Function Presentation and Print Services Facility

S544-5319-03

Preface

This publication provides information about using the IBM Print Services Facility feature of OS/400 (PSF/400) and is written for the general AFP user as well as for system and application programmers.

This publication helps you learn Advanced Function Presentation (AFP) and develop AFP applications. It parallels the AS/400 Advanced Function Presentation course and builds skills chapter by chapter. The first chapters set the stage for print and presentation on AS/400, covering the value of AFP applications, the AS/400 print flow, AS/400 AFP architecture and flow, and related applications. These chapters cover the two major aspects of AFP, the document architecture and the printing architecture.

“Chapter 5. Introduction to the Super Sun Seeds Case Study” on page 43 presents a sample invoicing application used throughout the publication. Subsequent chapters cover the building blocks of electronic publications: fonts, image and graphics, bar codes, and electronic forms. A series of “Using” chapters describes how to use different AS/400 application enablers to create the Super Sun Seeds invoicing application.

“Chapter 16. Using the Client Access/400 AFP Viewer” on page 233 shows how to view electronic applications, in addition to (or in place of) printing them. “Chapter 17. Using Facsimile Support, OfficeVision, and OnDemand for AS/400” on page 241 demonstrates how several AS/400 applications integrate with, and complement, the example AFP application. In addition, the publication includes chapters on network and cross-system considerations, and the appendices provide both additional examples and reference information.

Note: For more information about AFP for PSF/400, consult the IBM Printing Systems home page:

`http://www.printers.ibm.com`

Chapter 1. How AFP and PSF/400 Can Help You Present Information

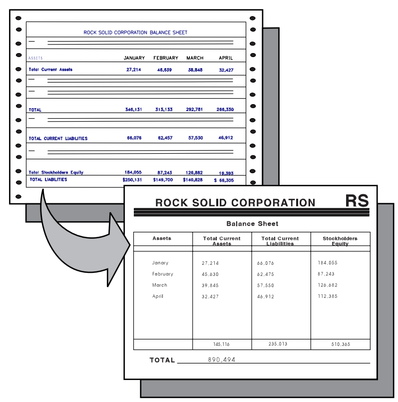
Advanced Function Presentation (AFP) is a printing and presentation system that enables you to take full advantage of printing technology on your AS/400 system. AFP helps you combine the capabilities of high-function Intelligent Print Data Stream (IPDS) printers and print software to:

- Create state-of-the-art documents
- Exploit print formatting capabilities without changing application programs
- Replace traditional, labor-intensive print operations with a system-managed process
- Print with complete system management and full error recovery, whether the printer is system-attached via twinax or network-attached via TCP/IP.

PSF/400 is the AFP system software for AS/400 printers using the IPDS printer data stream. PSF/400 enables AS/400 users and applications to take full advantage of IPDS printer capabilities.

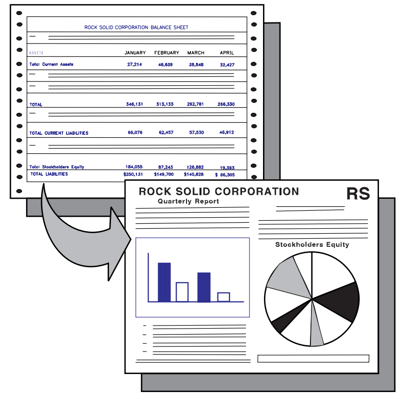
AFP provides a number of capabilities that can improve your current printing process. With AFP, you can:

Create your own forms in-house. This allows you to improve the flexibility and quality of your forms, avoid the expense of having someone create them for you, and avoid the inventory problems associated with preprinted forms (form obsolescence, storage, handling).



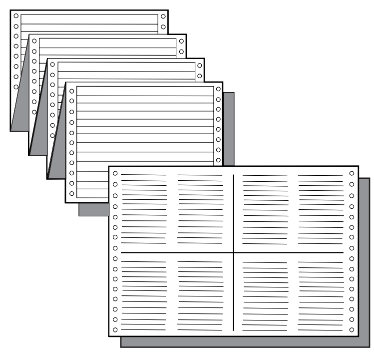
Create your own bar codes. This allows you to automate your business processes, improve data entry accuracy and speed, and meet industry requirements for bar codes.

Print line-mode data on page printers but also incorporate complex images and graphics such as graphs, charts, logos, and bar codes. This protects your investment in line-mode data and improves the quality and readability of your print output.

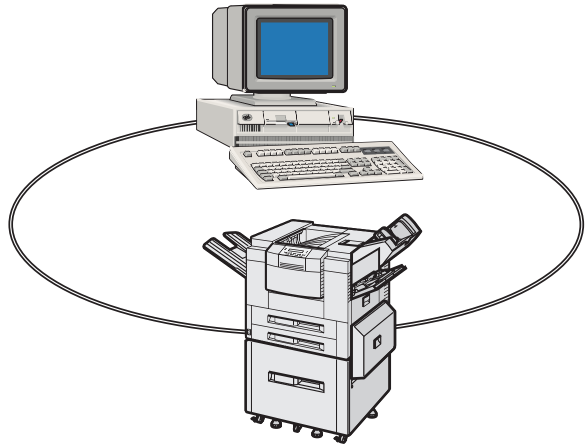


Save time, paper, and unnecessary printing expenses by viewing your formatted output, with graphics and text merged, on a computer display *before* printing it or *instead* of printing it.

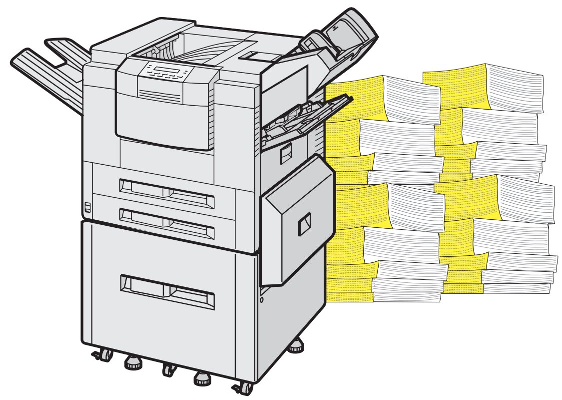
Save paper and storage space by printing your system output with smaller fonts and by automatically printing more than one page of data on a single sheet.



Print to network printers (i.e., TCP/IP-attached) with the same level of document function and print control that you have with system-attached, twinax printers.



Scale up applications from desktop print volumes to production print volumes without changes to applications.



What Are the Benefits of AFP?

Advanced Function Presentation (AFP) is an architected system for printing on AS/400 that integrates print management, application enablers, and high-function printers to generate high quality, effective documents that contain electronic forms, bar codes, images, graphics, and more.

AFP enhances the value of your output applications in the following ways.

Integration

The integration of application, development, and operational functions on AS/400 provides ease of use, productivity, and performance. AFP application development, among many enabling choices, uses high-level languages and Data Description Specifications (DDS), the same interface used in AS/400 application development.

Managing AFP output uses printer files and output queues, and is an extension of AS/400 print management. AFP is integrated into key AS/400 applications such as Imageplus/400, Officevision/400, Facsimile Support for AS/400, OnDemand for AS/400, and Backup Recovery and Media Services (BRMS) for AS/400.

Efficiency

The AFP data stream is a highly structured, architected data stream that yields efficient, high-performance results. Many elements of the electronic document, such as overlays, bar codes, fonts, and image, are managed as separate objects. This means that the actual data stream from the output application is usually small.

Print Management

AFP is integrated into AS/400 print management and provides robust functionality. This functionality allows flexibility in managing and printing output files. And, more importantly, print management tracks the printing process to ensure that print jobs are printed accurately and completely. PSF/400, because of the two-way communication between the system and the printer, can manage down to the page level. AFP provides complete error recovery, guaranteeing the successful delivery of print to the printer.

Network Printing

Placing printers on the network enables a variety of systems and applications to use them. However, the standard print model for network printers is TCP/IP, and native TCP/IP support for print is very limited. AS/400 uses IPDS to manage network printers attached with TCP/IP, achieving the same level of print function and print management as twinax-attached printers.

Object Management

AS/400 is an object-oriented system, as is AFP. This allows applications to be broken up into objects (such as programs, database files, printer files, display files) and developed or managed separately. AFP extends this structure to electronic document elements such as overlays, fonts, and images. Such a structure facilitates a high-performance print process. It also enables the AS/400 system to easily control access to print applications, thus providing a security function.

Printer Integration

The integration of high-function printers with AFP output means that the printing process can be optimized to efficiently handle the support objects for an electronic document. For example, images are compressed to minimize storage and download time while IPDS printers are optimized to process this image, resolving and printing it at high rated speeds. Printing resources, such as fonts, overlays, and images are managed across job boundaries in printer memory, thus improving overall job throughput.

Scalable Applications

Documents and applications are scalable from low to very high print volumes. AFP is a page- and object-oriented architecture that enables performance tuning of print applications as the volumes increase. The actual printing architecture, based on IPDS, enables the AS/400 and the printer to cooperatively manage the printing process. In addition, IBM AS/400 printers scale up from desktop impact printers at 375 characters per second to production laser printing systems at over 1000 pages per minute.

System Performance

The AFP print data stream, its integration into AS/400 spooling subsystems, and external print resources provide the tools to manage and optimize printing performance.

Print Flexibility

AFP applications are part of a broader set of printing requirements for an AS/400 environment, where different print streams and different types of printers are commonplace. A number of tools facilitate the “any to any” printing of application output. AFP print can be routed to network or client-attached printers. Network printing can be routed to AFP printers.

Host Print Transform provides AFP-to-PCL data stream transform services to print AFP output on PCL printers. An AFP-to-TIFF transform enables the use of 3489 graphical displays. Network print servers such as Print Services Facility for OS/2 (PSF/2) provide for the sharing and interchange of printers and print data streams.

Integrated Application Output

Data Description Specifications (DDS) provides powerful, effective electronic documents, because it is integrated with the application program. This means that an individual customer’s document can be precisely tailored based on the data for that customer. For example, a customer invoice can be designed dynamically, with invoice segments and document elements dynamically placed based on the actual content of the customer transaction. This results in more readable, effective documents.

Application-Independent Electronic Documents

Where application integration of electronic output does not fit, you have many choices to format your electronic documents, independent of the application program. Advanced Print Utility (APU), page and form definitions, Print Format Utility (a module of Advanced Function Printing Utilities for AS/400 or AFP Utilities), and many third-party products provide this capability.

Postscript and Image Processing

Network applications are generating document data streams, such as Postscript, and image formats, such as GIF and TIFF. Data in these formats can be converted to AFP and then stored or printed from the AS/400.

Beyond Printing

The AS/400 system provides full graphical viewing of print files through the integration of the AFP Workbench Viewer in Client Access/400. This view technology supports all types of print, fax, and image files. After you view a document, you can print, fax, or annotate it from your workstation.

Document and report output files can be merged with image data using ImagePlus, faxed outbound using Facsimile Support for AS/400, or indexed and archived for later retrieval using OnDemand for AS/400.

Chapter 2. Printing on the AS/400

AS/400 printing was formerly limited to a single print data format and was printed only on twinaxial-connected line printers. Print support in the AS/400 operating system consisted of a single printing subsystem that supported all print application output, spool management, and printer devices. Most AS/400 output was generated in the SNA Character String (SCS) print-data format and was printed on SCS line printers.

Breadth of AS/400 Printing Support

AS/400 now provides a wide range of printing support including AFP printers of various technologies, speeds, and capabilities. AS/400 has also extended print support to the network, enabling it to serve clients and printers on the network.

As printing technology rapidly evolved and organizational requirements for printing expanded, so did the printing support on AS/400. Initially, this expansion took the form of extensions to the SCS data stream (extended text attributes, imbedded image, for example) and support for PC-attached printers. Over time, additional printing support has addressed every phase of the print process, including:

- AFP and ASCII data stream support
- Broad choices in printer attachment, including network-attached printers
- Enhancements to print management, such as remote systems printing, print performance, and print workload balancing
- Extensions to printing, such as view, fax, and archive
- Support of network-attached TCP/IP printer using IPDS.

AS/400 Print Support Print Flow

The AS/400 print flow is shown in Figure 1 on page 13. The following sections describe this process.

An application program creates data to be printed. That output can be formatted in the program (program-described) or externally using DDS. Every program creating printed output on AS/400 has an associated printer file. The printer file defines many of the options of the printing process, from spool routing to page characteristics. Most output is spooled, that is, placed in a holding area called the output queue prior to printing. Actual printing is managed by the print writer, which sends the spooled file to the printer for printing. In fact, the print writer does far more. It must, for example:

- Retrieve the spooled file, and printer device description.
- Determine the print data stream to be sent, based on the device type in the printer file and printer definition in the device description.
- Convert the spool data stream as required.
- Manage the actual printer process. For AFP output, that means managing a two-way dialog with the printer to track each page through each station of the printer.

The components of the print process are described below.

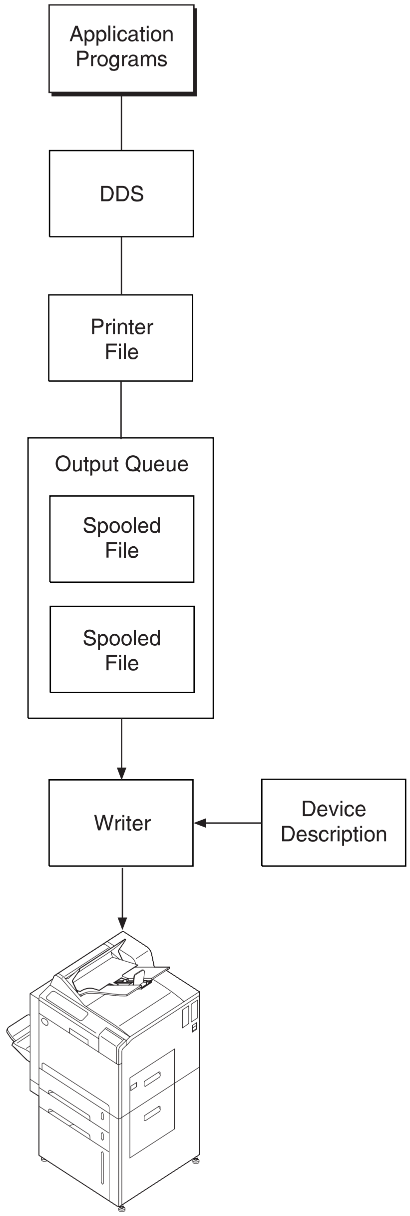


Figure 1. AS/400 Print Flow

Data description specifications (DDS) give you the ability to completely format your application output external to your application program. You can use DDS, for example, to include different electronic overlays on different pages or to select fonts or bar codes for different fields of text.

A **spooled file** is one that holds output data waiting to be processed. Being able to spool a file means that you can choose whether the processed printing job should go directly to a printer or be placed in an output queue where it can be stored until the printer is ready to print.

Printer device descriptions contain information that describes a particular device attached to the system. Device descriptions have to be created for each printer attached to the system. If you use automatic configuration, printer device descriptions are created by the system, with the exception of printers attached to an ASCII work station controller.

Optionally, you can use the Create Device Description (CRTDEVPRT) command to define each printer. You also can use the Create PSF Configuration (CRTPSFCFG) command to specify additional parameters for AFP printers that are not supported by the CRTDEVPRT command. Refer to *CL Reference* and *AS/400 Printer Device Programming* for more information.

The **printer writer** is a function of the operating system that writes (sends) the spooled file from the output queue to the printer.

When your job is ready to be spooled, you issue a call from your program to spool it. Before the call has been issued, though, you can still change or override the printer file (using the CHGPRTF and OVRPRTF CL commands).

After you have called your program, you can still do the following:

- Change some attributes of the spooled file (CHGSPLFA)
- Look at records in the spooled file (DSPSPLF)
- Check the spooled file status (WRKSPLFA)
- Check the printer status (WRKWTR)
- Display current overrides (DSPOVR)
- Look at a list of spooled files in a queue (WRKOUTQ)

AS/400 Print Data Streams

Printed output is the result of the interaction between the printer and the software that controls it. Different printed output requires different types of printers and different software (data streams) to control them.

A print data stream is a set of commands that tell a printer what to do. Some data streams, such as SNA Character String (SCS), include a limited number of commands. Others, such as Intelligent Printer Data Stream (IPDS), include a rich and complex set of commands. As a result, printers that support the simpler data stream are limited to the functions and capabilities of that data stream, whereas printers supporting the richer data stream offer more advanced capabilities. Therefore, it is important to know about data streams so that you can understand specific printer capabilities and limitations, as well as application compatibility and non-compatibility for various printers.

SNA Character String (SCS)

The SNA character string (SCS) data stream has a relatively simple structure, consisting of a one-byte hexadecimal code followed by the data to be printed. SCS, which is the standard pre-AFP print data stream, is used to control line printers and supports row and column functions. SCS and its superset Final Form Text Document Content Architecture (FFT DCA) also support the following printing functions:

- Underscores
- Overstrikes
- Emphasized text
- Partial line spacing
- Margins
- Superscript (with OfficeVision/400)
- Subscript (with OfficeVision/400)

- Font changes in a document (with OfficeVision/400)
- Symbols (with OfficeVision/400)

IBM AFP printers support SCS to provide compatibility with applications that generate SCS. See “Appendix A. IBM Printers and Compatibility Considerations” on page 283 for a list of IBM AFP printers that support SCS.

Final Form Text Document Content Architecture (FFT DCA)

FFT DCA is an extension of SCS used in the OfficeVision/400 environment to define how data streams that represent a document are to be printed and organized. FFT DCA is composed of one-byte and multi-byte (extended) control characters. FFT DCA controls the following functions:

- Top margin location
- Left margin location
- Line spacing
- Font definition
- Justification (aligning of text)
- Beginning and ending of underscores and overstrikes

IBM AFP printers support FFT DCA to provide compatibility with applications that generate SCS. See “Appendix A. IBM Printers and Compatibility Considerations” on page 283 for a list of IBM AFP printers that support FFT DCA.

Advanced Function Printing (AFP) Data Stream

Advanced Function Printing (AFP) Data Stream is the application interface to Advanced Function Presentation (AFP). AFP Data Stream includes data and text, and device independent (not printer specific) references to AFP resources, such as overlays, page segments, font objects, and so on. AFP Data Stream is independent of operating systems and page printers, is portable across environments, and is constructed of structured fields. AFP Data Stream output is produced by a number of application enablers, such as the AFP Toolbox, Advanced Print Utility, DDS, and so on.

AFP Data Stream is a companion data stream to the Intelligent Printer Data Stream (IPDS). The same family of IPDS printers is supported by both AFP Data Stream and IPDS. See “Appendix A. IBM Printers and Compatibility Considerations” on page 283 for a list of IBM AFP printers that support AFP Data Stream.

Intelligent Printer Data Stream (IPDS)

The Intelligent Printer Data Stream (IPDS) is a host-to-printer data stream for Advanced Function Printing subsystems. It provides an interface to all-points-addressable (APA) printers that makes possible the presentation of pages containing an architecturally unlimited mixture of different data types, such as high-quality text, images, graphics, and bar codes.

IPDS includes data and text, and resolved (printer specific) references to AFP resources, such as overlays, page segments, font object, and so on. IPDS supports an interactive, two-way dialogue between the print writer and the printer to provide:

- Printer information

- Cooperative error recovery
- AFP resource management (for example, managing font, image, and overlay resources in printer memory)

Note: IPDS processing depends on the specific printer data is sent to. Not all IPDS commands may be implemented in a specific printer.

The IPDS architecture is divided into functional areas called command sets or towers. This modular design allows printer developers to match selected command sets to specific needs. The command sets are:

- Device control (required)
- Text
- IM (raster) image
- IO (compressed) image
- Graphics
- Bar code
- Page segment
- Overlay
- Loaded font

IPDS is a companion data stream to the AFP Data Stream. The same family of IPDS printers is supported by both AFP Data Stream and IPDS. See “Appendix A. IBM Printers and Compatibility Considerations” on page 283 for a list of IBM AFP printers that support IPDS.

In addition to describing the composition of each page, IPDS is a bi-directional print management architecture. The device control commands facilitate a two-way dialog between the AS/400 writer and the printer. This dialog controls page development, job processing, job status, and error recovery. For example, at the start of a new print job, the writer will query the printer to determine what font, image, and overlay resources are already in printer memory. Any missing resources are sent down to the printer. Each operation is managed to completion (i.e., an acknowledgment that a resource was received successfully).

PostScript to AFP Data Stream

PostScript is a data stream generated by many PC and network station applications. The new IBM Network Station, which can be attached to an AS/400 server, generates the PostScript data stream. If that PostScript data stream is being spooled to the AS/400, the data stream must be converted to AFP data stream in order to print on an IPDS printer. PSF V4R2 supports the PostScript-to-AFP Data Stream conversion for PostScript level 1.

To facilitate this conversion from PostScript to AFP data stream, the system administrator must configure the AS/400 device description with the IMGCFG parameter, setting it to the name of the IMGCFG object that describes the output device. See ″Using an AFP Printer to Print PostScript Output″ in Chapter 19 for a list of limitations and details regarding the device configuration necessary to enable PostScript to AFP data stream conversion.

American National Standard Code for Information Interchange (ASCII)

Unlike the other printer data streams covered in this section, no formal structure governs the use of the ASCII data stream to control printers. Although ASCII forms the basis for the control and character codes, not all ASCII printers conform fully to this standard. Additional command sets based on hexadecimal codes are often used with advanced function printers to exploit the capabilities of these printers. In the ASCII environment, therefore, the range and types of printing functions available to the user are governed by individual programs tailored to suit each type of printer. See “Appendix A. IBM Printers and Compatibility Considerations” on page 283 for a list of IBM AFP printers that support ASCII.

ASCII printer applications are printer dependent, and ASCII printers are driven by emulator programs that create an ASCII set of controls from an EBCDIC data stream.

While most ASCII printer data streams have been created to support specific printers, two standards have emerged for describing pages of ASCII information: Postscript and PCL. Postscript is a standard administered by Adobe. PCL is administered by Hewlett-Packard.

In AS/400 printing, ASCII support is provided by translating SCS and AFP print data streams to their ASCII equivalents. This translation is normally done by Host Print Transform (it can also be done at the client through print emulators). ASCII printers can be attached to AS/400 in a variety of ways, and all can use Host Print Transform. Host Print Transform is integrated into the following printing functions:

- AS/400 Print Writer
- Send TCP Spooled File (LPR)
- LAN Print Driver

Host Print Transform is specified in the printer device description. A large number of ASCII printers (each with its own set of ASCII printer control codes) can be selected. Host Print Transform can provide two types of data stream transformations:

- SCS to ASCII
- AFP to PCL or PPDS

SCS to ASCII

Host Print Transform translates SCS to ASCII, thereby providing 3812 SCS printer emulation. This is similar to the printer emulation in Client Access/400 Workstation Function or Rumba/400.

AFP to PCL or PPDS

Host Print Transform translates full-page AFP output into HP PCL or Lexmark PPDS print data streams. This transform uses one of two different modes: raster or mapping, depending on output and printer characteristics. Raster mode converts the AFP pages to image. This preserves print fidelity, but at a cost of print file size and performance. Mapping mode converts AFP resources (fonts, bar codes, overlays, and page segments) into PCL and PPDS equivalents. Refer to “Chapter 18. Network Printing” on page 261 for more information on AFP transforms using Host Print Transform.

The AFP to PCL/PPDS transform expands your choices when implementing AFP printing applications. It is particularly well-suited for providing the capability of reprinting subsets of a larger AFP printing job (reprinting an invoice, for example) or supporting smaller print volumes at a remote branch office. Factors such as print fidelity, AS/400 system impact, and overall print performance can affect how this function is used.

COBOL, you do not have to be concerned with the structured fields because OS/400 creates them for you.

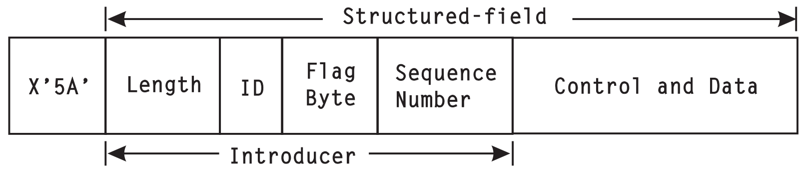


Figure 3. Structured Field

AFP is also a printing or presentation architecture. It is based on the MO:DCA-P (Mixed Object: Document Content Architecture-Presentation) standard, which is both system and printer independent. As the name implies, the architecture defines how mixed objects (image, graphics, fonts, overlays, and bar codes) are to be pulled together and presented as a single page or document. The structure of a MO:DCA-P document is shown in Figure 4.

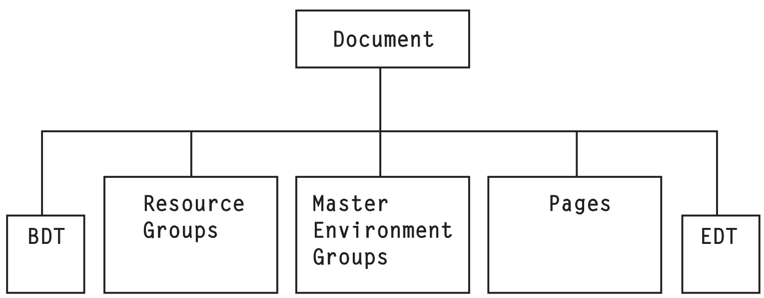


Figure 4. MO:DCA-P Document Structure

As Figure 4 illustrates, every AFP document is framed with “Begin Document (BDT)” and “End Document (EDT)” structured fields. After the Begin Document, resources to be used, such as overlays, page segments, or fonts, are defined. Then, the standard or default options, such as fonts, for the entire document are defined. And lastly, the actual pages of the electronic output are specified.

MO:DCA-P provides a precise structure to page and document composition. Consider the Super Sun Seeds invoicing application. If the document is defined as one thousand Super Sun Seeds invoices, the structure (and resulting performance and efficiency) emerges. Page elements (fonts, overlays, and images or page segments) are defined up front in the resource group. These elements are defined once and then referenced as required throughout the one thousand invoices. The master environment group defines characteristics about groups of pages (for example, copies of a page) or sub-pages. Again, basic composition structure is defined once and used throughout the document. The pages section contains the specific structured fields to build each page.

MO:DCA-P consists of mixed objects, with each of the constituent objects having its own structure or architecture. The object types are described in the following sections.

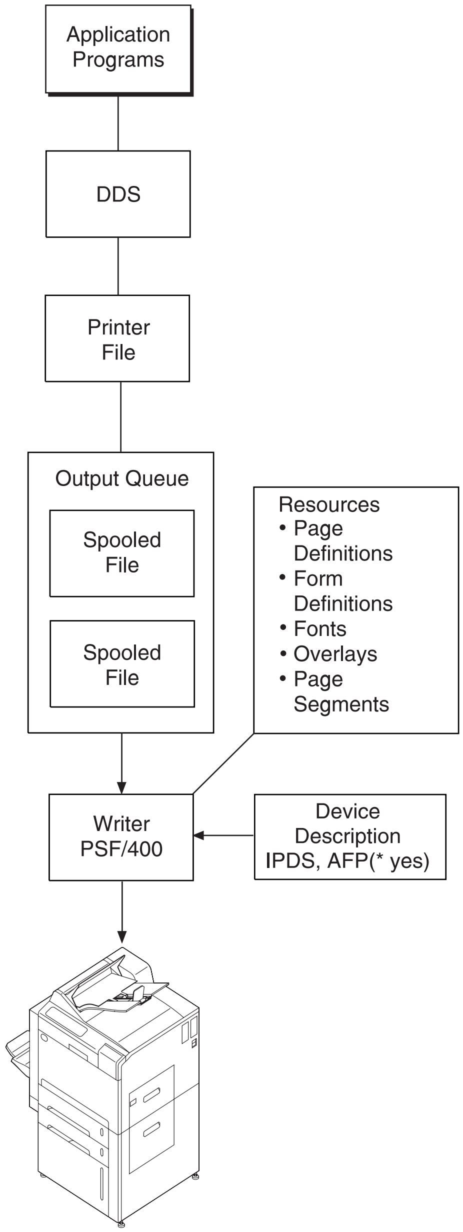


Figure 6. AFP (PSF/400) Printing Subsystem Print Flow

The AFP (PSF/400) printer subsystem print flow is described below.

Application programs generate output that can be either program-described or externally-described. If the output is program-described, then the application program (RPG, COBOL, and so on) formats all of the data on the page one line at a time. The program can produce AFP directly, or it can use an application enabler such as the AFP Toolbox to create the output.

If the output is externally-described, the application program simply creates records. These records are processed by Data Description Specifications (DDS), which is separate source and object code from the RPG or COBOL program.

DDS keywords determine how each field of every record from the application program is formatted on the page. Fields can be individually placed on the page in

any font or orientation. Some fields can be placed on the page as bar codes. DDS keywords allow lines and boxes to be drawn, as well as many sophisticated formatting options, such as the selection of AFP overlays. DDS can be used in both the OS/400 printer subsystem and the AFP printer subsystem, but many more formatting capabilities are available with AFP. See “Chapter 11. Using Data Description Specifications” on page 127 for more information.

The application program may or may not use DDS, but it always uses a **printer file**. This is the same printer file used in the OS/400 printer subsystem. Values specified in the printer file determine general characteristics of the entire print job, such as what printer the job will be sent to, whether to print simplex or duplex, whether to print multiple pages per side, whether to add an overlay to every page, and so on.

A **spooled file** is created on an output queue, just as it is in the OS/400 print flow. The difference is that an AFP spooled file can contain references to resources that will be needed to print the file. These references result from the printer file parameters and DDS keywords that were used when the file was created. The resources may be downloadable fonts, AFP overlays, and page segments (scanned images such as logos or signatures). Other resources that may be referenced by a spooled file are page definitions and form definitions, even though they are less common in the AS/400 environment. See “Chapter 13. Using Page Definitions and Form Definitions” on page 193 for more information.

In the AFP print flow, the **device description** of the printer to which the output is being sent will specify a device type of IPDS, AFP(*YES). This means that the printer is an IPDS device capable of handling all AFP functions, such as downloading resources, printing graphical information from DDS, and so on.

The function of the **print writer** in the AFP print subsystem is performed by Print Services Facility for AS/400 (PSF/400), which is a feature of the operating system.

The PSF/400 printer writer has three primary functions:

1. Perform any necessary data stream transforms on the spooled file and take it off the output queue. The data stream of the spooled file may be either SCS, IPDS, or AFP data stream, as determined by the printer file. The spooled file must be transformed into the correct printer-specific IPDS data stream.
2. Gather any resources referenced by the spooled file. These resources can be fonts, page segments, overlays, and so on.
3. Send the transformed spooled file and its resources to the printer. Because IPDS is a bidirectional data stream, the print writer must also participate in a two-way dialogue with the printer. Through this dialogue, PSF/400 manages error recovery. If printer intervention is required during the printing of a job, for example, PSF/400 resumes printing at the next page in the job so that no data is lost and the job does not have to be sent to the printer again.

Printers and PSF/400

PSF/400 supports the entire family of IBM IPDS printers as well as IPDS printers from other vendors. PSF/400 support may be required or optional, depending upon the particular printer specified.

Figure 13 shows a sample electronic form (overlay).

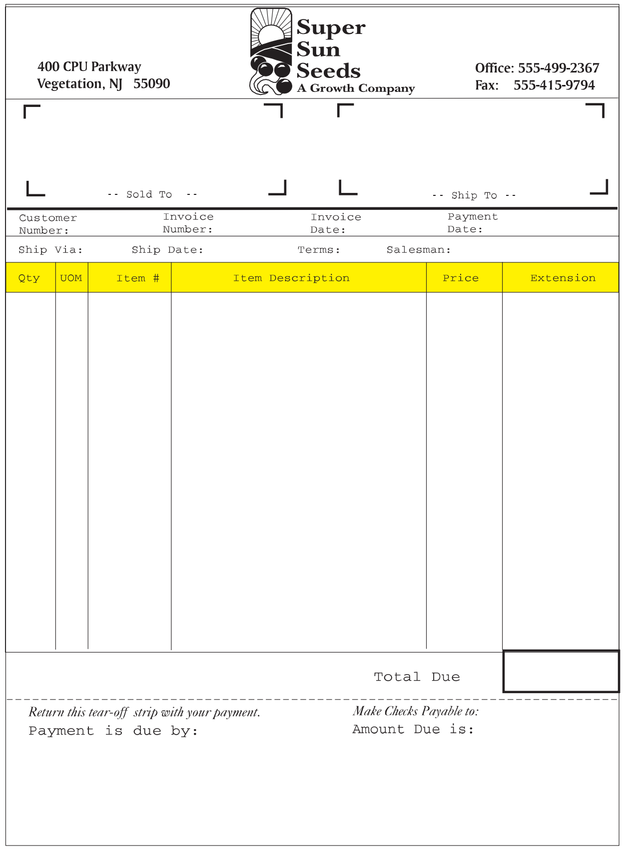


Figure 13. Sample Electronic Form (Overlay)

AS/400 AFP Formatting Resources

AFP supports two external resources for formatting pages—form definitions and page definitions. These resources contain rules for mapping application output into fully composed pages. Page definitions and form definitions are external to the application program and thus separate the final document formatting from the application. Since these resources represent a subset of the functionality available with DDS, you would normally use them where you wanted the formatting to be application-independent.

Form Definitions

A *form definition* is the resource that specifies the physical attributes of the printed output. The word *form* refers to a sheet of paper or other print media.

Support for creating and using form definitions in the AS/400 AFP subsystem was added in PSF/400 V3R2. Before that, form definitions could be created only on the RISC/6000 and S/390 systems or with PC products such as ISIS or Elixir, and could be used on the AS/400 by means of a set of APIs called PrintManager/400, or by routing print jobs from MVS or VM using Network Job Entry (NJE).

A form definition performs many of the same functions as a printer file on the AS/400, but whereas a printer file is required, a form definition is optional. If you do

A0260

not specify a form definition, media handling characteristics are taken from other existing keywords. If you do specify a form definition, its media handling instructions override those specified in existing keywords.

If you choose to use a form definition, you must specify its name in the printer file.

A form definition contains printing controls that specify the following, within the limitations of each printer:

- Page origin (the top-left boundary for printing). Page origin may be different for the front and back of the page when duplexing.
- Sheets that have overlays printed on them.
- Number of copies of each page to be printed.
- Paper source (input bin of the printer) for printers with more than one paper source.
- Simplex (printing on just one side of a sheet) or duplex (printing on both sides of a sheet on printers that support duplex printing).
- Page presentation (portrait or landscape).
- Print-quality level (on printers that support different levels of print quality).
- Horizontal adjustment in pels or dots per inch (dpi).
- N_UP Printing: printing multiple logical pages on a sheet. These pages can be either MO:DCA-P pages (fully composed pages containing data and the structured fields controlling presentation of the data) or line data.

See “Chapter 13. Using Page Definitions and Form Definitions” on page 193 for more information on form definitions.

Page Definitions

A *page definition* is the resource that specifies how line data is to be formatted into pages by PSF/400. PSF/400 does not use a page definition for MO:DCA-P (AFPDS) data, because that data is already composed into pages before PSF/400 receives it.

A page definition performs many of the same functions that are available on the AS/400 by means of DDS. The page definition is an alternative to DDS for formatting data on a page independent of the application program.

Specifying a page definition is optional. If you do not specify a page definition, media handling characteristics are taken from existing keywords. If you do specify a page definition, its media handling instructions override those specified in existing keywords.

If you choose to use a page definition, you must specify its name in the printer file.

A page definition contains formatting information that specifies the following:

- Fonts to be used for printing the data
- Where data from each input record is to be printed
- Constant data to be printed
- Data fields that can be suppressed
- Print position for carriage controls or channel codes
- List of page segments

Chapter 11. Using Data Description Specifications

Using Data Description Specifications (DDS), which is included in OS/400, for application output enables you to take full advantage of the advanced printing capabilities of AS/400 in much the same way as using external database files enables access to the advanced database capabilities of the system. DDS printer file support provides full control of output, and supports the advanced electronic printing capabilities of today's laser printers. DDS gives you complete control over each page and over all the elements that come together on a page.

DDS support for advanced electronic business documents builds on existing support for line-mode output. DDS supports standard document composition elements, such as electronic forms, images, graphics, bar codes, lines, and boxes. Direct control is provided, for example, over what electronic forms or images go on to what pages. Support for full control of document pages and page layout from the application program also is provided. In addition, nearly all DDS support is dynamic; that is, the application controls both the element and its positioning, enabling you to produce documents with "floating" page elements.

As described in "Chapter 10. Using Printer Files" on page 115, every AS/400 job has a printer file that provides page defaults such as margins, line spacing, and overflow.

The printer file also provides printer and queue options such as printer device, file type, and spooled file save. It provides page composition, such as front and back overlays and page and form definitions. Generally, these parameters apply to the entire job or spooled file.

Printer File DDS

DDS printer file support externalizes application output and extends it to full-page applications. Output that is defined within the program is called ″program-described″ output. Output that is defined with DDS is called ″externally-described″ output.

Figure 61 on page 128 shows an example of how DDS is used within the printer file to define application output. The program builds the data fields to be printed and does a write to a DDS record format. The data fields are referenced within the record format. The printer file, through the DDS keywords, controls the position, orientation, font, and other characteristics for those fields. In addition, DDS provides access to all the elements—text, overlays, images, graphics, bar coding, lines, and boxes—that comprise AFP documents.

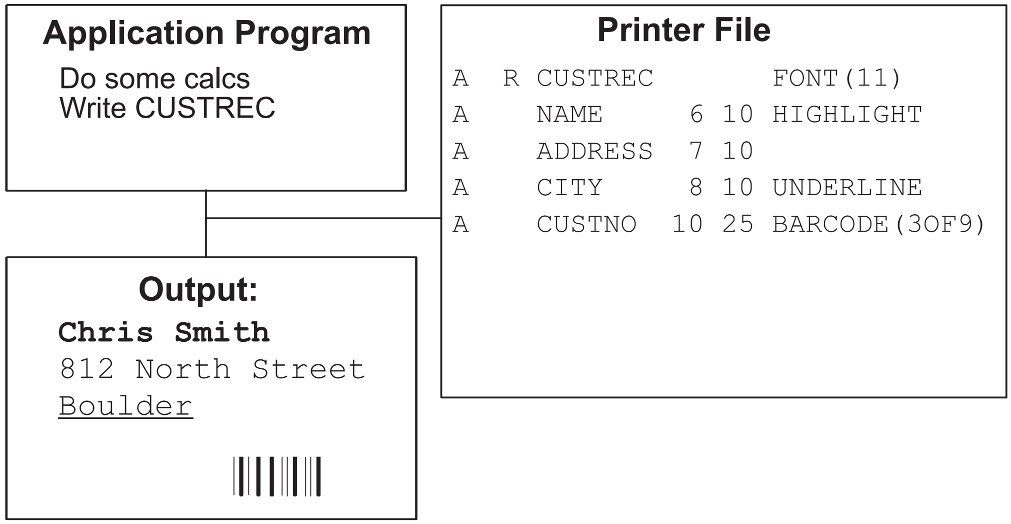


Figure 61. Externally Described Printer File with DDS

Figure 61 shows a simple example of how DDS print formatting works. The application program prepares the variable data. It also establishes the logic of printing, or in this case, the ″writes″ to DDS records. A DDS print record is the collection of fields and/or print keywords that are to be executed when the application program issues the write command. Once the record write and variable data is passed to DDS, DDS can control font, positioning, and other characteristics external to the application program.

DDS groups one or more individual fields together to create a record. The application program controls printed output by deciding when to write which records.

Some DDS keywords apply only to the entire record (record-level). Other DDS keywords apply only to fields (field-level). Some DDS keywords can be used at either record-level or field-level.

Keywords for AFP Applications

The keywords described in the following sections are used to enable applications using full page mode. Keywords for page layout and for page composition are described.

Page Layout Keywords

The following sections describe keywords used for page layout.

LPI

LPI is a record-level keyword used to change lines per inch. When the LPI keyword is specified, it overrides the LPI parameter of the printer file.

The format of the LPI keyword is:

```
LPI ( 4 6 8 9 12)
```

The format of the COLOR keyword is:

```
COLOR (BLK BLU BRN GRN PNK RED TRQ YLW)
```

TXTRTT

TXTRTT is a field-level keyword used to rotate fields.

The format of the TXTRTT keyword is:

```
TXTRTT (0 90 180 270)
```

HIGHLIGHT

The HIGHLIGHT keyword prints a field in bold characters. The active font for the field must be a numeric font (FONT keyword) that supports bold printing.

The format of the HIGHLIGHT keyword is:

```
HIGHLIGHT
```

UNDERLINE

The UNDERLINE keyword underlines a field.

The format of the UNDERLINE keyword is:

```
UNDERLINE
```

A summary of all printer file DDS keywords can be found in “Appendix F. Data Description Specifications (DDS) Reference” on page 329. Refer to *AS/400 Data Description Specifications Reference* for more complete information on the keywords described in this section.

Finishing Operation Keywords

Table 3 on page 126 and Table 4 on page 126 show how PSF/400 programming support for finishing operations relates to various releases.

ZFOLD

The ZFOLD record-level keyword causes the current sheet to be first folded in half inwards (so the front side of the sheet is now inside the fold) along a line parallel to the reference edge. The half of the sheet furthest from the reference edge is again folded in half outwards along a line parallel to the reference edge. For example, when applied to an 11″ by 17″ sheet with the reference edge along a short side, the result is an 8.5″ by 11″ fold-out.

The format of the ZFOLD keyword is:

```
ZFOLD (reference-edge paper-type)
```

The reference edge indicates the edge to be used as a reference for the first fold. The paper type can be ledger or A3.

Figure 62 on page 135 illustrates Z-folded sheets within an edge-stitched document.

Finisher Output

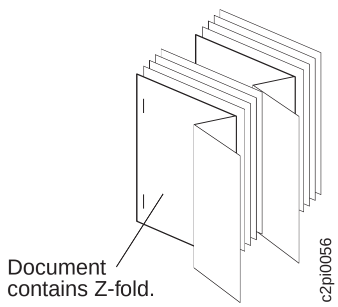


Figure 62. Edge-Stitched Document with Z-Folded Sheets

A summary of all printer file DDS keywords can be found in “Appendix F. Data Description Specifications (DDS) Reference” on page 329. Refer to *AS/400 Data Description Specifications Reference* for more complete information on the keywords described in this section.

DDS Functions

Now that the DDS keywords used to enable full page composition have been introduced, we can show how these keywords are used in a sample case. Figure 63 on page 136 shows how DDS can be used to produce multiple printing function output.

Chapter 13. Using Page Definitions and Form Definitions

When we look at DDS as an enabler for documents and reports, the integration of formatting with the application program was a significant advantage. It provides the application programmer with the capability to produce very customized output conditioned by the database and application data within the program.

However, there are environments where this tight integration is less desirable. It can make the task of coding application logic and output logic more complex because the logic is intertwined. Furthermore, the developer who works with the application programs may be different from the designer/developer that works with output formatting. There is a trade-off here: the customized output that DDS provides versus separation in the application development process.

New output formatting objects on the AS/400, page definitions and form definitions, provide a means to separate page formatting from the application program. These new objects, an industry standard in high-volume printing environments, were added to OS/400 with Version 3 Release 2 and Version 3 Release 7. The manner in which these formatting objects were implemented provides a dynamic method of transforming existing line-mode (SCS) print applications.

As illustrated in Figure 116 on page 194, an SCS print application sends lines of output to the output queue. With the simple reference of the page definition and form definition objects in the AS/400 printer file (and no changes to the application program), the output produced in the output queue is no longer SCS, but is AFP.

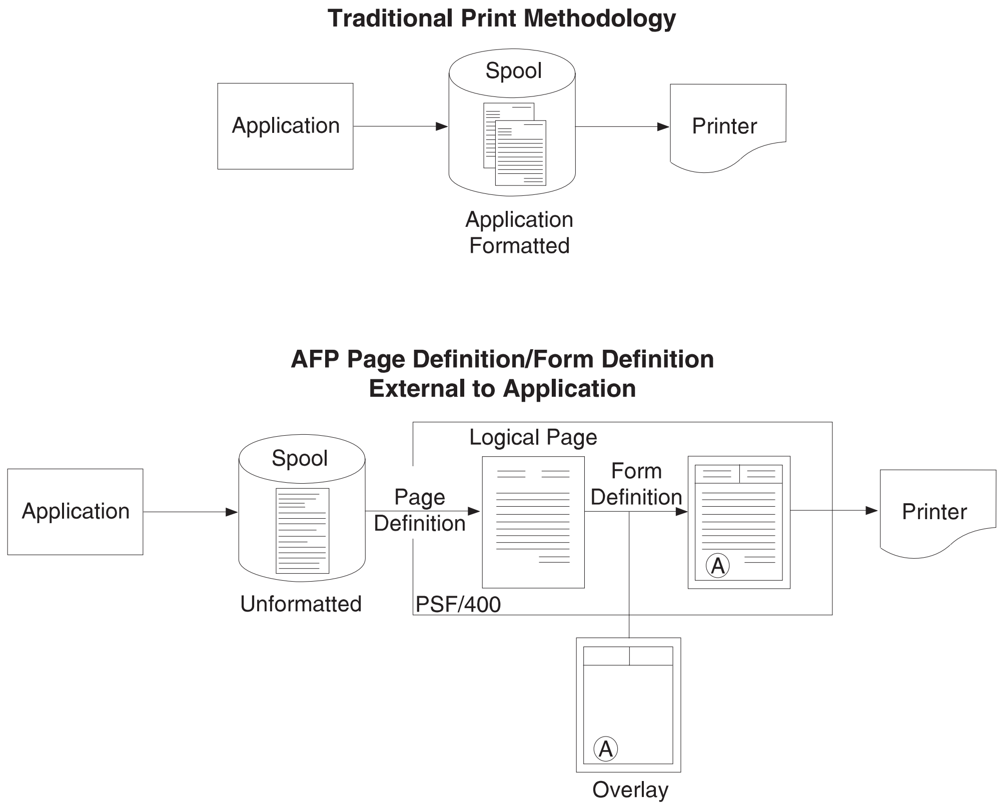


Figure 116. Traditional Printing Compared to Printing with Page Definitions and Form Definitions

Page definitions and form definitions look and act like page formatting programs. They are developed in a source language (which can be front-ended with several different graphical tools) that defines how information is to be placed on the page. Specifically, the page definition defines how data is placed on a logical page layout. Input print lines are read in, optionally parsed into individual fields, and place on the page. Similar in structure to DDS, the page definition language enables you to place print lines or print fields anywhere on the page while controlling font, orientation, and color characteristics. Data can also be printed in a selected bar code symbology. Figure 117 on page 195 shows a conceptual view of the function of the page definition.

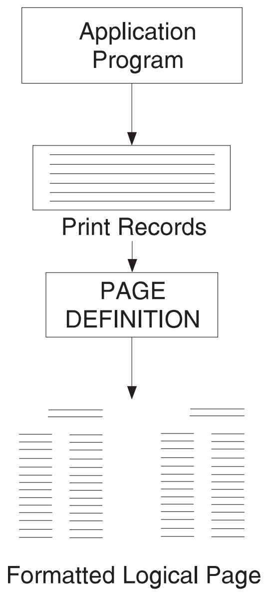


Figure 117. How a Page Definition is Used

Again like a program, page definitions can contain conditional logic. This means that the formatting rules can change based on the contents of an input field. A ″trigger″ field (for example, company number) may be used to select a whole series of formatting commands. The form definition is normally used in conjunction with the page definition (it can, however, be used by itself).

The form definition controls how the logical page (defined with the page definition) is placed on the physical medium - the sheet of paper. Source statements within the form definition specify what drawer paper is selected from, what overlay(s) is placed on top of the logical page layout, whether duplexing is used, whether multiple logical pages should be placed on a single physical page, what copies are to be automatically created, and what fields should be suppressed from which copy. Figure 118 on page 196 illustrates the form definition concept.

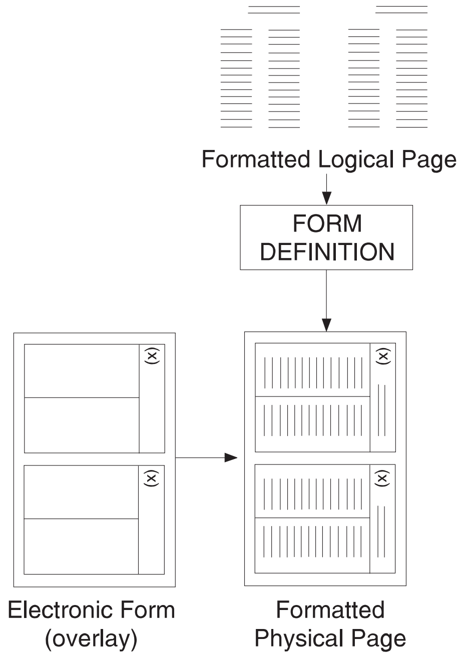


Figure 118. How a Form Definition is Used

An additional note should be made on the use of page and form definitions. In addition to specifying the form definition or page definition name in the printer file, the output file type must also be changed to *LINE. This indicates that the print lines will be passed to the page and form definition objects in a slightly different format than SCS. This format is called line data, and it adds blank lines (automatically) for vertical positioning and a couple of additional carriage control characters to each print record. Again, this is transparent to the application program. To the designer/developer of the page definition, however, they must be aware that space and skip to lines will be passed to the page definition as blank lines. They can either be printed as blank lines (to maintain vertical positioning) or thrown away, whichever is appropriate to a particular page layout.

How can you use Page Definitions and Form Definitions

1. Preparation stage
   a. The application generates line data.
   b. The person responsible for designing the output layout creates a page definition and form definition to format the line data into the desired form. This step may require a detailed knowledge of the application-generated line data.
2. Production stage
   a. The application generates line data specifying the corresponding page definition and form definition resources through new parameters on the CRTPRTF and OVRPRTF commands.
   b. The line data is sent to an AFP(*YES) defined printer where PSF/400 uses the page definition and form definition specifications to create a formatted data stream which is sent to the printer.

   Note: page definitions and form definitions can either be created directly by the customer using Page Printer Formatting Aid (PPFA) or obtained

a file copy. To implement this application, a change or override to the invoicing printer file is made to include *LINE as the device type and SSSFD1 as the form definition.

SSSPD1—Example of a Page Definition

The page definition reads and formats the actual print line data. Line data differs from SCS print data in that line data contains all of the print lines, including the spacing and skipping lines. In the following example, we show page definition SSSPD1, a simple reformatting of the invoicing print lines.

```
 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
 /* Pagedef Name: SSSPD1                                              */
 /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
PAGEDEF SSSPD1               /* SSSPD1 is the page definition name   */
   REPLACE YES;              /* PPFA should replace existing copy    */
                             /* Define font nicknames as follows:    */
   FONT HELV                 /* HELV is the short name               */
         CS H200A0           /* C0H200A0 is character set            */
         CP V10037;          /* T1V10037 is code page                */
   FONT BOLD 440000;         /* Courier Roman Bold 10 Char/Inch      */
   FONT ITAL 4300B0;         /* Courier Italic Normal 12 Char/inch   */
   FONT NORM 420000;         /* Courier Roman  Normal 10 Char/inch   */
   SETUNITS 1 IN 1 IN        /* This indicates that the default unit */
                             /* for all measurements is the inch     */
     LINESP 6 LPI;           /* Lines spacing at 6 lines/inch        */

    /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
    /* All print lines handled by this one page format               */
    /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

    PAGEFORMAT 1OF1;         /* 1OF1 is page format name             */

    /* Customer Address Block - first 14 lines of page               */
    PRINTLINE
      CHANNEL 1              /* When an input line has a first col.  */
                             /* (CC) '1' value, this PRINTLINE gets  */
                             /* control - meaning: top of a new page */
      POSITION 0 0           /* Horiz Pos:0     ; Vert Pos:Down 0"   */
      FONT BOLD              /* Use bold font                        */
      REPEAT 14;             /* Process 14 lines this way            */

    PRINTLINE                /* Last line of name and address
      POSITION 0 NEXT        /* Horiz Pos:0;      Vert Pos:Next line */
      FONT BOLD;             /* Bold font                            */
      FIELD START 16         /* Parse zip code field out for bar     */
            LENGTH 5         /* code                                 */
      POSITION MARGIN -1 IN  /* Postnet bar code at page margin, 1"  */
            BARCODE          /* up from last line position           */
            TYPE POSTNET;    /*                                      */

    /*                                                               */
    PRINTLINE                /* Next 35 lines of page                */
      POSITION 0 NEXT        /* Horiz Pos:Margin; Vert Pos:Next line */
      FONT NORM              /* Normal font                          */
      REPEAT 35;

    /*                                                               */
    PRINTLINE                /* Total due line
      POSITION 0 NEXT        /* Horiz Pos:Margin; Vert Pos: Down Next*/
      FONT BOLD;             /* Bold font                            */

    /*                                                               */
    PRINTLINE                /* Last 12 lines of page                */
```

```
POSITION 0 NEXT       /* Horiz Pos:Margin; Vert Pos:Next Line */
FONT NORM             /* Normal font to be used here          */
REPEAT 12;            /*                                      */
```

This page definition reads and prints each of the print lines with the following changes:

- Invoice name and address is printed in bold.
- The zip code field is selected and reprinted in bar code.
- The total line is printed in bold.

Looking at the SSSPD1 page definition, the heading information defines the fonts and line spacing to be used. Note that the prefix ″C0″ will be added in front of the character set name and the prefix ″T1″ will be added in front of the code page name. For the font name assigned HELV, the character set C0H200A0 will be used. This set is Helvetica in Roman Medium typeface with a point size of eleven.

This page definition contains one page format. All print lines will be processed by this format. Each input print line maps to a PRINTLINE statement. The first PRINTLINE has a CHANNEL keyword. This means that this PRINTLINE statement will assume control when the input print lines advance to the top of a new page (including the first page). This application has 66 print lines per page, including spacing and skipping, but this doesn't require 66 PRINTLINE statements. The REPEAT keyword enables multiple, successive print lines to be handled by the same PRINTLINE statement.

The first PRINTLINE statement handles the first 14 print lines. This includes the initial 12 blank lines and the first two print lines containing the invoice name and address. These lines are printed in the BOLD font, which is defined as 10-point Courier Bold and is found in character set C0440000. The next PRINTLINE statement handles the name and address print line that contains the US zip code. This print line is also printed in Courier Bold. However, the print line is then parsed (subdefined into individual fields) to define the zip code field. This field is then printed in US Postnet bar code. This bar code is printed at the current horizontal position (MARGIN), but placed one inch above the current vertical position. Therefore, the bar code will print directly above the name and address data just printed.

The remaining PRINTLINE statements handle the rest of the print lines down the page. All of these lines, except for the total due line at line number 51, are printed in the NORM font. The total due line is printed in the font defined as BOLD.

To implement this page definition, the invoice printer file would be changed or overridden to specify *LINE as the device type and SSSPD1 as the page definition name. If the previously described form definition (SSSFD1) was also used, then you would end up with three copies per input page with the line formatting handled by the page definition and the overlays applied as defined by the form definition.

Additional Form Definition and Page Definition Formatting Functions

The page and form definition contained in “Appendix L. Page Definition and Form Definition Source Code for Super Sun Seeds” on page 367 provides for far more formatting function than the page and form definitions just described. Based on the input invoice data, one of the following three output page types would be created:

- Single page invoice
- First or interior page of multi-page invoice

- Last, or summary, page of a multi-page invoice

In order to create the output page type, conditional logic must be defined. Within the page definition (SEEDS), a PRINTLINE statement is parsed in order to look for the words ″seeds″, ″trees″, and ″fruits″ that are found in the ″Thank you″ message found at the end of each invoice. Locating these words means that we are on the last page of an invoice. In addition, these words are also used as a trigger to place an image of a seed, tree, or fruit on that page.

The page definition has three different PAGEFORMAT records corresponding to the three different output page types. Based on the conditional logic, control is switched dynamically to the correct PAGEFORMAT record. Control passes back to the first PAGEFORMAT record at the top of a new input page (because it contains the CHANNEL keyword). Note that the form definition contains seven copy groups, corresponding to the output page type and which image (seed, tree, or fruit) is required. The selection of the right copy group is defined within the page definition.

Printer File Keyword Support when Using Page Definitions and Form Definitions

With the introduction of line data, the AS/400 can receive layout and media handling instructions through existing Printer File Command keywords and from new page definition and form definition output-descriptors. You must, therefore, describe which keywords are ignored when a page definition or form definition is specified and which keywords are used when no page definition or form definition is specified.

Print File Keywords Ignored when Line Data Is Specified and a Page Definition Is Used

The following printer file keywords are ignored when line data is specified and a page definition is used:

- CDEFNT
- CHRID
- CPI
- FOLD
- FONT
- FNTCHRSET
- LPI
- LVLCHK
- MULTIUP
- PAGESIZE
- PAGRTT
- REDUCE

Print File Keywords Ignored when Line Data Is Specified and a Form Definition Is Used

The following printer file keywords are ignored when line data is specified and a form definition is used:

- BACKMGN
- DRAWER (if *FORMDF is specified)
- DUPLEX (if *FORMDF is specified)

Overlays and Page Segments

Viewing AS/400 Spooled Files with Client Access/400

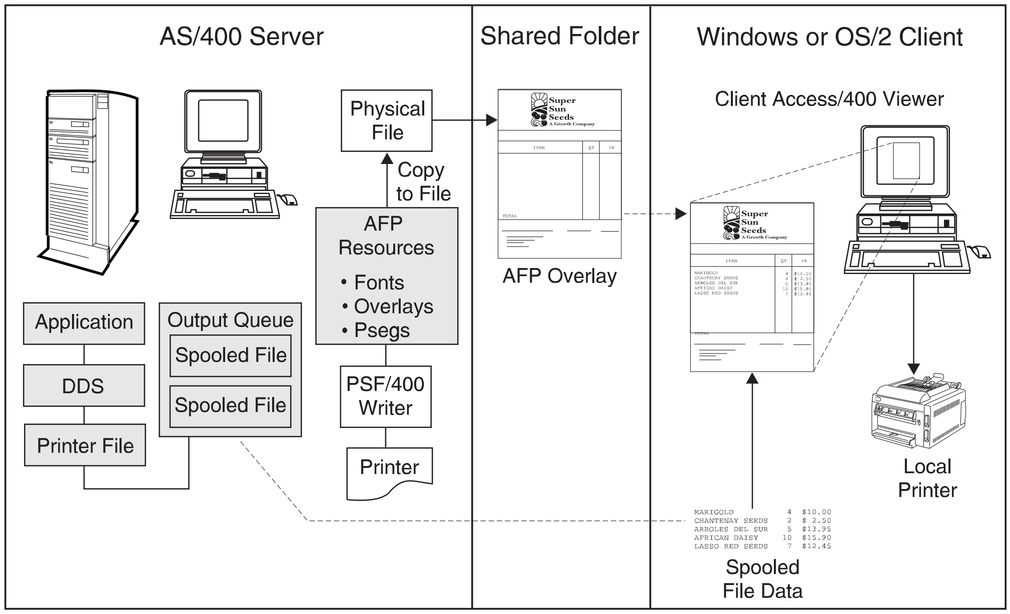


Figure 140. Viewing AS/400 Output Using Client Access/400

In order to understand this process, consider the case study of the Super Sun Seeds Invoice. When that job is a spooled file in an output queue, it contains references to page segments (Super Sun Seeds logo, and others like the strawberry, tree, and flower). It also contains references to several different AFP overlays (the different variations of the invoice form). These resources are not “inline” with the spooled file; they are simply referenced by the spooled file. If this job is released to an IPDS, AFP(*YES) printer, PSF/400 gathers all the resources and sends them to the printer along with the spooled file.

For Windows 95 and Windows NT clients of Client Access, the spooled file is sent along with the externally-referenced overlays and page segments. For OS/2 and Windows 3.1 clients, these resources must be available in a shared folder or PC file directory. If these resources are not already available within a shared folder or PC file directory, they must be copied there. Detailed instructions for copying overlays and page segments can be found in “Appendix C. Setting Up the Client Access/400 Viewer to View AFP Resources” on page 309.

Fonts

The AFP Workbench Viewer uses PC-resident fonts to display documents. This means that the font used in the AS/400 document will be matched as closely as possible with a font available on the PC when the document is displayed with the AFP Workbench Viewer.

By default, the PC uses True Type fonts, which are included with Windows. If the PC also has Adobe Type Manager (ATM) fonts installed and active, the AFP Workbench Viewer uses these fonts to display documents. ATM must be purchased separately from the AFP Workbench Viewer.

Chapter 18. Network Printing

AS/400 printing traditionally meant printing lines of print records on twinax-connected impact printers. Much has changed - electronic printing applications, printer technology, LANs and LAN applications, the extended computing network.

The emergence of LANs in AS/400 customer locations has created two separate print environments: AS/400 host-centric printing of SCS or AFP output and LAN printing of PC-generated output, usually in a PCL or PostScript format. More and more AS/400 customers are looking for ways to integrate these two separate environments. Figure 172 takes a look at AS/400 requirements from an applications perspective.

	AS/400 Legacy Applications	AS/400 AFP Applications	LAN Applications
Output	Lines of text	Sophisticated documents with fonts, images, graphics, bar codes, electronic forms...	Sophisticated documents with fonts, images, graphics, bar codes, electronic forms...
Data Stream	SCS	AFPDS, IPDS	PCL, PostScript
Bi-Directional for error recovery	Yes	Yes	No (though bi-di ascii is emerging)
Print Technology	Mostly impact; some laser	Mostly laser; some impact	Almost all laser;
Printer speed ranges	Up to 2,200 lpm or >729 ppm	Up to 1,200 lpm or >729 ppm	Typically up to 24 ppm, can be up to 40 or 50 ppm

Figure 172. SCS, AFP, and LAN Printing Applications

Traditional AS/400 output is lines of text and/or data in SCS printed a line at a time on mostly impact printers. AFP applications are sophisticated pages of output in first AFP (system) then IPDS (printer) format printed on mostly laser printers. The AS/400 natively supports only AFP and SCS directly. ASCII is supported through transforms of AFP and SCS, or via an individual application (which is transparent to the AS/400).

As the AS/400 is a multi-user, data processing system, the output applications tend to be line-of-business with significant page volumes.

LAN applications also produce sophisticated pages of electronic output, most in either PCL or Postscript format. LAN applications tend to be personal with lower volumes.

Many AS/400 customers have applications in all three categories and want the flexibility to handle the output with AS/400 print management and common printers.

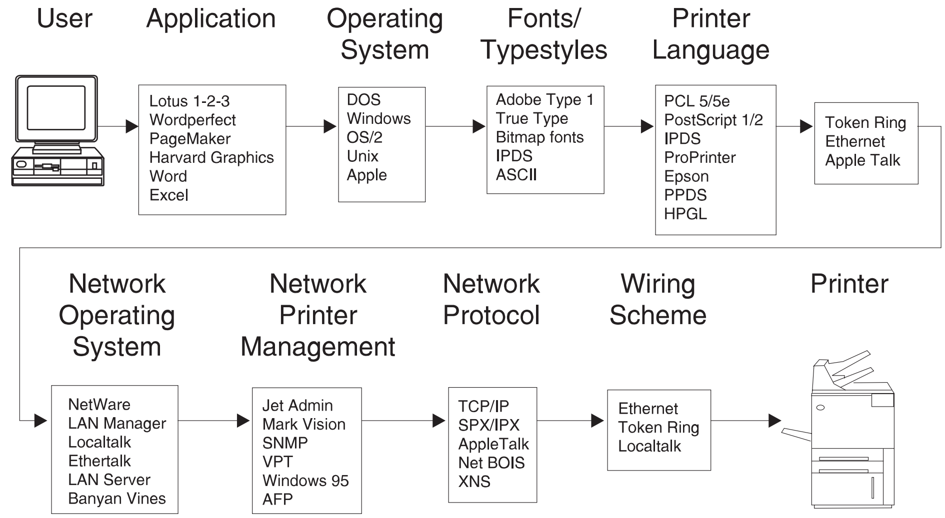


Figure 173. Network Printing Flow

Figure 173 takes a closer look at the network printing flow, revealing the complexity in moving output from application to printer. Applications work under different operating systems and use different resources and printer languages to compose pages. Once composed, print files may be managed by different operating systems and print managers. They may also move across the network, using different communication protocols (TCP/IP, SNA) and different LAN topologies (Token Ring, Ethernet).

In a network-centric environment, TCP/IP is the standard communications protocol. TCP/IP has rapidly become a standard within AS/400 installations. AS/400 network printing enables print files to be moved from AS/400 to network, from network to AS/400, and from AS/400 to other systems. We will look at a number of these network printing scenarios.

In many network print environments, print data must also be transformed. For SCS or AFP print files, the following perform conversion to ASCII formats:

- Host Print Transform
  - SCS to ASCII
  - AFP to PCL
- Advanced Print Services—Warp Server (formerly Print Services Facility for OS/2 or PSF/2)
- Print Services Facility for AIX
- Client Access/400 and Rumba print emulators

For Postscript and PCL print files, transforms are performed by:

- Client Access/400 Virtual Print
- Network Print Server (via APIs)
- Print Services Facility for AIX

Host Print Transform is covered in more detail at the end of this chapter.

A0486

AS/400 Output to LAN Printers

Let's look first at moving AS/400 print files to LAN-connected printers. Figure 174 shows AS/400 print and attachment internals and a simplified LAN structure.

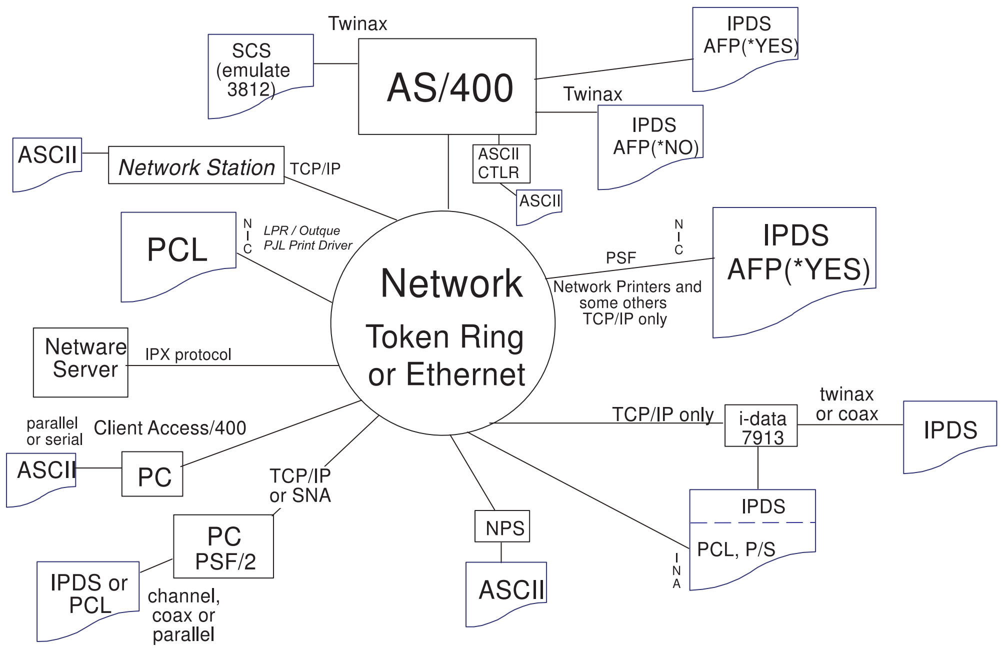


Figure 174. AS/400 to LAN printing

Output files can be sent to:

- LAN-attached IPDS printers directly, with complete print management
- IPDS or PCL printers attached to Warp Server
- Client Access/400-attached ASCII printers
- ASCII printers attached to Non-Programmable Terminals (NPT)
- Printers managed by the Netware integrated server
- Other printers using Network Print Server APIs.

LAN-attached IPDS Printer

Until Version 3 of OS/400, printing to a system-attached twinax printer and printing to a network-attached printer were two very different propositions. With system-attached printers printing SCS and IPDS applications, you had printer file functionality, interactive print process management, and complete error recovery. With network-attached printers, much of this control and function was missing. Standard TCP/IP printing is done through a simple file transfer called Line Printer Requestor (LPR). Most of the AS/400 printer file function and all of the print management control is missing. The spooled file is simply sent to an IP address. An IPDS connection applies an interactive, bi-directional print protocol to this environment. Except for auto-configuration, LAN-attached IPDS printers function the same as system-attached printers. Figure 175 on page 264 illustrates the conceptual flow from AS/400 (using PSF/400) to the LAN-attached IPDS printer.

A0487

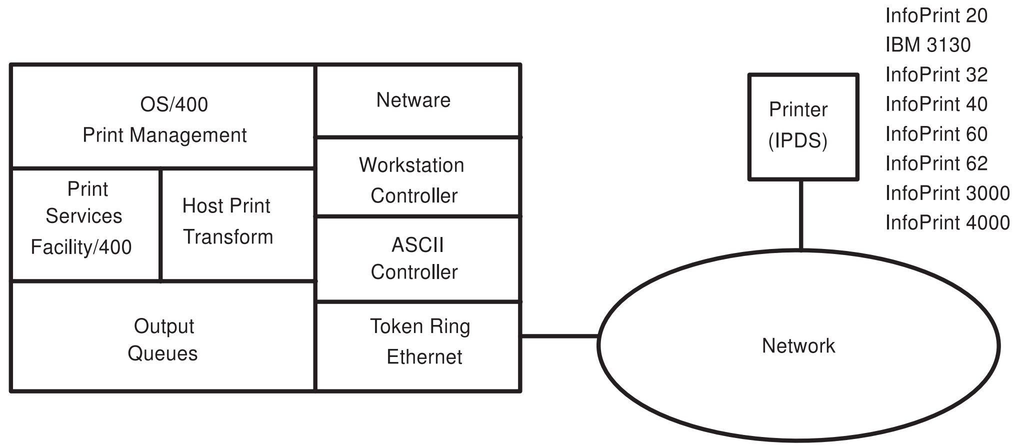


Figure 175. LAN-attached IPDS Printing.

LAN Print Servers

LAN print servers can receive print jobs from AS/400 and manage those files to either IPDS or ASCII printers. IBM LAN print servers include:

- InfoPrint Manager for AIX
- Print Services Facility for AIX (PSF/AIX)
- Advanced Print Services for OS/2 (formerly Print Services Facility/2, or PSF/2)

Printing can be asynchronous or synchronous. With an asynchronous connection, print files are transferred from AS/400 and handled independently by the LAN print server. With a synchronous connection, called Print Services Facility Direct or PSF Direct, printing passes through the LAN print server directly to the target printer. The printers defined to and attached to the LAN print server appear to AS/400 as direct-attached printers.

Distributed Print Function (DPF)

DPF is used to connect an AS/400 to a printer driven by Advanced Print Services. DPF can simultaneously route printed output from several AS/400 systems to different PC-attached printers. In DPF, each printer has a unique device definition on the PC workstation and each connection between an AS/400 and a printer has a unique DPF host receiver. The host receiver is responsible for spooling each AS/400 print file to the PC workstation spool. When the print file and print resources have arrived at the PC workstation, the DPF host receiver sends them to the printer queue. From the DPF printer queue, the print file is sent to the connected LAN print server printer.

The combinations of AS/400s and printers supported is limited by the number of DPF sessions. A session is defined as a connection between a host and a printer, with the printer being equivalent to a host receiver. You can have up to ten AS/400s connected to the same printer, one AS/400 connected to 10 printers, or any combination of hosts and printers that equals ten.

A0488

Figure 176 shows one example of a basic network structure for moving client print applications to the AS/400.

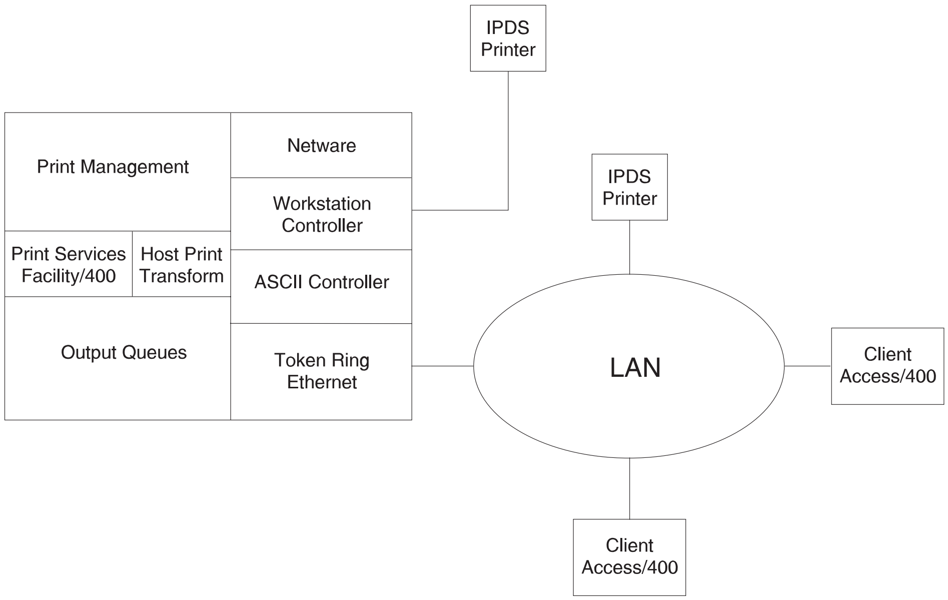


Figure 176. Printing Client applications on AS/400 printers

For full-page client applications (using Postscript or PCL), the print data stream must be converted into IPDS. Figure 177 shows how ASCII print from a Windows application flows to an AS/400-connected IPDS printer. The IBM AFP Printer Driver for Windows is used to generate AFP from the PC application, rather than an ASCII data stream like PCL or PostScript. This process is transparent to the user. The PC print job is automatically intercepted by Virtual Print and re-directed to an AS/400 Output Queue associated with an IPDS printer.

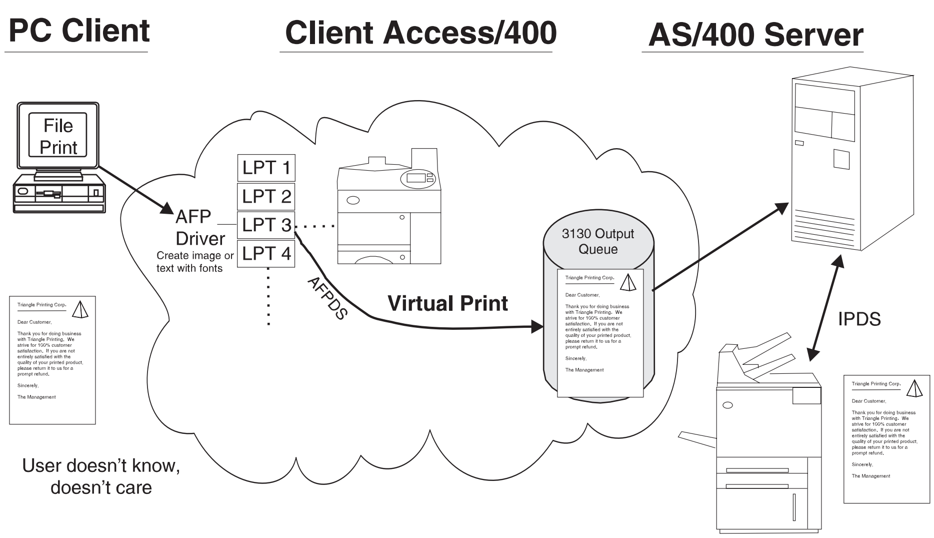


Figure 177. Printing PC applications on AS/400 IPDS printers

Network Print Server

The Network Print Server is the AS/400 component that handles virtual printing. The network print functions used with virtual printing are also available via APIs (Application Program Interfaces). These APIs enable client/server applications to invoke network print function.

LAN Output to LAN Printers

Providing SCS, AFP, and ASCII print support in a single printer does not necessarily require using print transform services. Some printers now provide all of these capabilities. These are workgroup or departmental printers, and they feature multiple attachments to multiple print servers (including the AS/400) while handling multiple print data streams.

The IBM 3130 and the IBM Network Printers are examples of workgroup printers with these capabilities. They can be simultaneously connected to multiple systems (up to three). For example, they can be twinax connected to the AS/400 and Ethernet connected to the LAN. The printers will dynamically switch between print jobs sent by the connected systems. With a feature called automatic sensing, these printers determine which print data stream has been received and the print data stream is correctly resolved. Figure 178 shows a typical multi-connection environment for a workgroup printer.

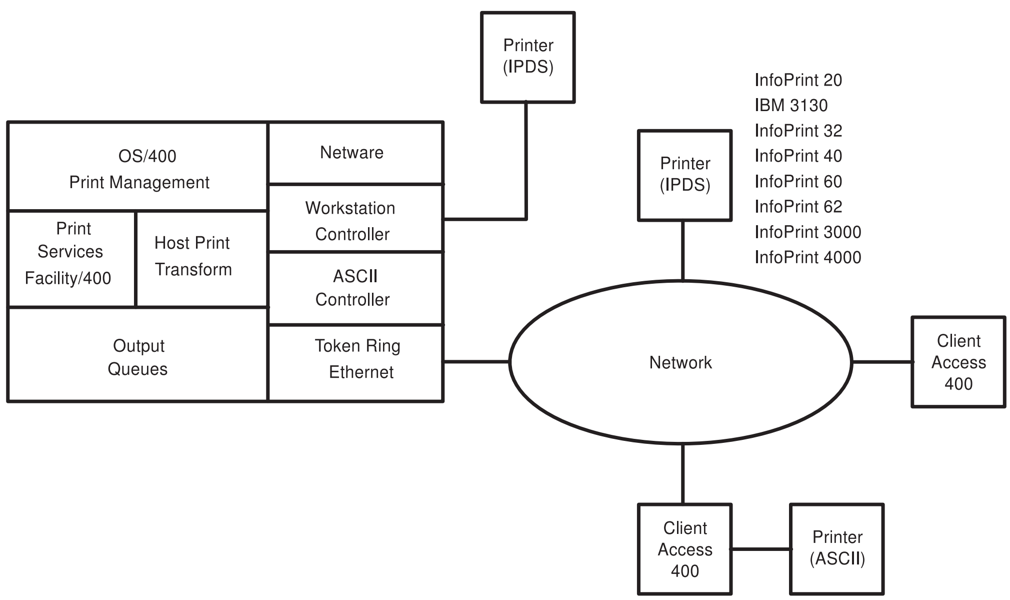


Figure 178. Printing LAN applications on workgroup printers

AS/400 Output to Other Systems' Printers

Network printing can also mean moving AS/400 print files to other AS/400s, to other AFP systems, and to other systems in general. Remote output queue and printer pass-thru enable one AS/400 to treat queues and printers on another AS/400 as if they were local resources. Once the connection is established, you have complete AS/400 print functionality.

AFP print files can be sent to other AFP systems. All required AFP resources (i.e., fonts, overlays, page segments) must be on the target system in order to print.

Chapter 19. Cross-System Printing

In many work environments today, printing across networked computer systems is a daily requirement. As an architected, structured print system, AFP can help you print to and from IBM and non-IBM systems. The same printing subsystem runs in all IBM operating systems, and data stream transforms are available in some of these to transform common data streams from other systems. AS/400 further helps printing on remote systems through the Start Remote Writer (STRRMTWTR) command, which enables spooled output files to be sent automatically to other systems using SNA Distribution Services (SNADS) or Transmission Control Protocol/Internet Protocol (TCP/IP).

Cross-System Printing Overview

Cross-system printing involves transmitting print requests from one system to another.

On AS/400, the creation of print files is similar to that on other platforms, but the implementation is different. AS/400 uses a printer file to define the attributes of the printed output, and the spooled file, with attributes, is passed to the target system.

Other systems cannot process the AS/400 printer file but may map some or all attributes specified in the printer file to corresponding attributes in the AS/400 system. The architected AFP resource objects (form definition, overlays, fonts, and images) can be processed in all operating systems. Page definitions can be processed in all AFP environments except OS/2 with PSF/2, which requires additional business partner software.

PSF/400 is included as a chargeable feature on AS/400 as part of the OS/400 operating system. PSF/400 is similar to PSF in other operating systems, although the interfaces to it are different in each system.

Supported Data Streams and Formats

AS/400 applications can produce spooled files in the following data streams:

- SCS (SNA Character String)
- IPDS (Intelligent Printer Data Stream)
- AFP Data Stream (Advanced Function Printing Data Stream)
- LINE (Line-mode data stream, or IBM 1403 printer data stream on S/370)
- AFPDSLINE (Line-mode data stream mixed with AFP data stream)
- USERASCII (ASCII data stream, including PostScript)

  An ASCII data stream, including PostScript, can also be placed in a spooled file, in which case the device type is USERASCII. Such a spooled file could be created by a PC or Network Station application and spooled to AS/400 by the virtual print function of PC Support/400. You also can create or purchase AS/400 applications that build their own data stream in ASCII.

  The type of data stream produced is determined by the DEVTYPE parameter of the printer file associated with the application.

  If you intend to send print files to another system, you should limit your use of DEVTYPE to data streams that can be printed on the target system or that can be transformed by network software programs.

ELECTRONIC PUBLISHING, VOL. 7(2), 75–87 (JUNE 1994)

Interleaf active documents

PAUL M. ENGLISH AND RAMAN TENNETI

Interleaf, Inc.
9 Hillside Ave
Waltham, MA 02154, USA

email: pme@ileaf.com, raman@ileaf.com

SUMMARY

A commercial structured document processing system has been built with an extensible object system. This system is an excellent platform for the design, implementation, and delivery of active documents. Examples are discussed.

KEY WORDS Active documents Document-based applications User interfaces Document-object systems Lisp

1 INTRODUCTION

1.1 Motivation for building an active document system

Interleaf 6 (I6) is the document creation component of Interleaf's document management software. It was originally designed for the creation of technical manuals that could be hundreds of thousands of pages in length. Such complex documents influenced the software's requirements, not just in performance and quantity handling, but also in its ability to address several organization and production workflow issues regarding documentation. One of the things done to address such requirements was to build a highly extensible system—one that could be customized to fit the specific needs of a particular site. One goal of this extensibility when it was designed in 1989 was the ability to create *active documents* and document-based applications.

1.2 Motivation for building active documents; the document user interface

Paper documents and books are the most common user interface (UI) for information presentation. Most people are more comfortable using paper documents than using a computer. Just by looking at a book most people can quickly recognize titles, paragraphs, headings, sections, figures, lists, and so on. There are well understood (even if not always followed) UI and organization standards for paper documents and books. They usually follow the same organization, with a title page, copyright and publisher information, contents, chapters, and an index.

Because documents are so familiar, it follows that more document-based computer applications are being built. The currently most popular personal accounting software,

CCC 0894–3982/94/020075–13 *Received 5 April 1993*
Revised 12 January 1994

Quicken by Intuit, does not have a generalized graphical user interface (GUI)—it instead has a UI which simulates a paper check. Use of this natural UI makes this product easy to learn and hence extremely popular. Likewise, applications built with the document metaphor are easy to learn for most people.

1.3 Outline

An architectural overview of I6 is presented in Section 2, followed by a description of related work in Section 3. Sections 4 thru 7 contain descriptions of four applications of active documents constructed with I6, including interesting active document issues with each. Sections 8 thru 10 compare active document solutions with other solutions, discuss some implementation issues, and conclude with some of the main points presented.

2 ARCHITECTURAL OVERVIEW

2.1 Document structure

I6 is an object-oriented structured document editor. The main structure is called a *component*, which contains content as well as formatting information. For example, in the I6 representation of this paper, the current paragraph is a *para* component. The numbered section title is a *head* component, which contains an *autonumber token* from the *autonumber stream* named *list* as part of the *head* component's *prefix*. A component prefix is a special case of an *inline* component, which is used structurally to set off content that is formatted *in line* with its parent component. Inline components are used to represent nested structure, and often have different style attributes from the containing parent. I6 also has font and other style tokens that can be placed in the middle of a component text stream to change style without requiring an inline component. *Reference* objects exist to allow references that share content (usually an autonumber or page number) with other document objects.

Frames are containers used explicitly to control the placement of content including text and graphic objects such as *diagrams*, *images*, *charts*, *equations*, etc. The author creates a frame within a component. One of the formatting properties of a frame controls its location relative to its *token* (creation point) within its parent component. Expressing frame location relative to its anchor token allows greater layout flexibility than requiring the user to place the frame on a specific place on a specific page. For example, when content is inserted before the frame, the frame *knows* to move with its anchor to the next page. Each page has special *header* and *footer frames* set to have repeating content.

Tables are composed of *rows*, which are a special kind of component. Table cells are then special cases of frames, and thus can contain anything contained by a frame.

Microdocuments are component containers that can live inside frames.

2.2 Implementation architecture; extension language

Most of I6 is implemented in C. The heart of I6 is its object system, which has a dynamic class hierarchy, message set, and method associations per class. The object system includes features such as multiple inheritance, property inheritance, and *before* and *after* methods [1].

I6 also has a tightly integrated Lisp system, which is based on Common Lisp [2]. The I6

developer's environment includes an editor, listener, compiler, debugger, syntax checker, help system, a number of other utilities, and the entire source set for the Lisp portion of the system. In addition to the simple Lisp editor, GNU Emacs [3] has been customized thru its own Lisp environment to connect to running I6 sessions to edit and debug I6 Lisp, and to provide interactive I6 help.

Lisp scripts can be attached to any Interleaf object, stored within or external to a document. End-users need not know or care that the system contains Lisp, since generally they never see it.

The I6 UI is written entirely in Lisp. (There are Lisp bindings to a Windows-enriched Motif toolkit based on OSF Motif 2.0 and Microsoft Windows. This allows sharing over 95% of the UI code between Motif and Windows. The I6 Macintosh product has Lisp bindings to the Mac Toolbox, used to build the I6 Macintosh UI.) Callbacks are written in Lisp, although they usually simply send messages into *editor objects*, which are mostly written in C.

Lisp was chosen as the I6 extension language because it is interpreted (providing a fast prototyping environment) and has run-time binding. Run-time binding is required for easy delivery of active documents. This allows a user to simply cut and paste a document from an active system to a static one, then allowing the active document to introduce new classes and methods into the previously static environment. (See Section 9 for a discussion of activation and security issues.) Other features of Lisp that have proved useful include its exception handling, list processing, automatic memory management, packaging, and ability to be rebound to create problem-specific sub-languages.

3 RELATED WORK

The most intensive and well-documented research effort in active document technology has taken place at Xerox PARC. Spinrad [4] seems to have originated the term *active document*. Zellweger's Scripted Documents [5], Arnon's CaminoReal [6], Bier's and Goodisman's embedded buttons [7] and Active Tioga documents [8] were built using Tioga, Cedar's multimedia document editor [9]. The Tioga editor allows tagging of the nodes of a document tree and tagging of individual characters. In CaminoReal documents, editing of a mathematical object can send related mathematical objects to a "standard mathematical system," returning other objects that in turn are reformatted into the document. Zellweger's Scripted Documents contain sequences of activities in the form of external scripts to support such things as voice animation. Bier and Goodisman describe a prototype architecture where arbitrary document elements behave as buttons (control elements). Active Tioga [8] allows procedures written in the Cedar programming language and registered in a central database to be invoked as activities in response to document editing or redisplay.

Other active document research efforts have taken place at IBM, Apple, and the Weizman Institute of Science. Chamberlin et al. [10] describe the Quill document editor, which allows programmers to attach procedures written in the REXX programming language to individual objects of a document. While Quill used these procedures primarily for formatting purposes, it is clear that they could be used to add activity more generally to documents. Towner [11] describes a tool that imports database text fields into a document by adding special markup into the document. In this auto-updating tool, the activity—content updating—takes place only at a user's explicit request. Hansen [12] describes the Ness component of the Andrew ToolKit, which allows authors of documents to attach

scripts that are written in the Ness programming language. These scripts are triggered by top level user events invoked via the mouse or keyboard. Goldberg et al. [13] add activity to ordinary email in order to facilitate the use of email for collaborative work.

Interleaf's active document architecture is described by English et al. [14], who define active documents as "structured documents and their processors in which the objects in the documents can be acted upon by, and can themselves act upon, other objects in the document or the outside world."

4 EXAMPLE: AUTO-LOCALIZING TEMPLATES

I6 is distributed with a number of document templates—sample memos, letters, reports, etc. These templates usually contain some initial text and definitions of the master component types (sometimes called *style sheets*). I6 needs to be localized for fifteen languages and over forty countries, and this work includes the localization of the document templates. The memo template should initialize with page size 8.5″ by 11″ in the USA, but page size A4 in the UK. And the *para* component should be called *Absatz* in Germany.

In prior releases of the system, template maintenance was time-consuming and error-prone, as each change to a template required someone to edit the corresponding copy for each of the other languages and countries. I6 contains a system to allow the template designer (a typographer or graphic artist) to build *auto-localizing templates*.

The template documents contain tables of localization information, including the names of all components as well as values for localized variables such as page size. See Figure 1.

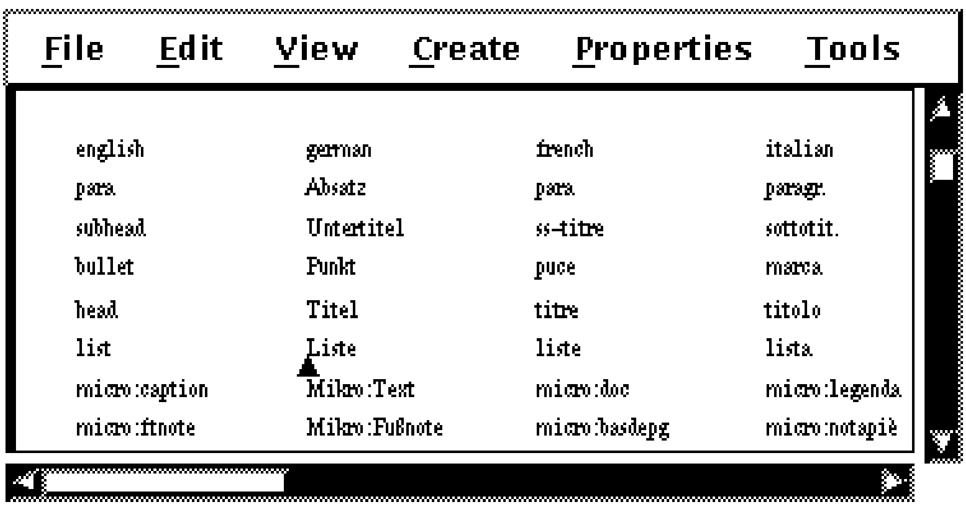


Figure 1. A template localization table

A template document subclass is defined in one of the system Lisp libraries. This subclass has a custom *open* method that changes component names and sets other document properties according to the table-specified settings for the current language. This occurs while the document is opening, but before its content is shown. The open method then purges the localization table, and changes the class of the document back to the default

document class, thus permanently deactivating this newly created document instance. This leaves its initial content in a state suitable for user completion.

The initial implementation of auto-localizing templates revealed general problems relating to confusion of building documents with building document *builders*. One problem is that active documents are often self-modifying, and thus one must be careful to save a copy of the original document. Another problem is that there is no consistent way to switch documents between static and active modes. This is exacerbated in this case since the last act of the subclassed open method is to turn the document back to the default (static) document class. Similar problems have been described in debugging Active Tioga documents [8].

Another problem with the template documents was a performance penalty users paid in order to have such run-time localization flexibility. On slower machines this added almost one second to the opening of a new document created from a template. At the cost of run-time localization flexibility, a site administrator can, at installation time, force the localization of the template documents in one instance for each language. This flushes the auto-localization templates and deactivates the documents, effectively freezing the template translations. This has proved acceptable because run-time localization is only important at a multilingual site; most sites are unilingual.

5 EXAMPLE: MULTIMEDIA EMPLOYEE DATABASE

We are presently building an active document employee photo album that is programmatically generated and has subclassed objects with custom selection methods. This builds in some ways on the active telephone directory described in [8] and the Biography Retrieval application described in [15]. The photo album currently has three types of data for each employee:

- Color image: a 35mm color photograph processed onto and extracted from a Kodak Photo CD.
- Voice: employee name as copied from our voice mail system.
- Text: information such as username, phone extension, and workstation name.

A Lisp program generates the photo album document from the above data. Each generated page contains a frame with a face image, a text description about that person, and a button component labelled with the user's workstation name. The face frames and the button component objects are subclassed, with new *select* methods provided for each. Clicking on the face plays the audio name, and clicking in the button component produces a list of processes being run on the remote workstation.

The code below defines the new frame subclass used to contain face images. A new select method is provided, which opens (plays) the audio object named by the current user.

```
(defvar face-class (obj-new-class doc-frame-class nil))
(mid:provide-method face-class mid:select #'face-select)
(defun face-select (frame &rest args)
  (audio-play-name (album-current-name)))
```

The code below defines the new component subclass used for the workstation machine name buttons, which on selection output a list of active processes. This `mach-select`

method calls `tell-next` to forward the select message on to other select methods that might exist for the `mach-class` subclass or its parent classes.

```
(defvar mach-class (obj-new-class doc-cmpn-class nil))
(mid:provide-method mach-class mid:select #'mach-check)
(mid:provide-method mach-class mid:enter #'mach-check)
(defun mach-check (cmpn &rest args)
  (mach-ps-display (mid:get-substring cmpn t t))
  (tell-next args))
```

The object system allows interception of events on a specific object instance or on a class-specific set of objects. Thus there is no need to intercept mouse button events, nor the system-wide *select* method, nor even the default system *select* method for frame objects. The `face-class` frame subclass was created to have the special `face-select` method, allowing selection of only the face frames being handled by special code.

It is also possible to provide *instance methods* directly to the special objects instead of having to subclass them. But in cases where there are many special objects (such as face frames above), it is more efficient to create a subclass with the new method (putting all desired frames in this subclass) than it is by providing instance methods for all instances of interest.

6 EXAMPLE: INTELLIGENT MAINTENANCE AID

Intelligent Maintenance Aid (IMA) is an active document and expert system that was developed by United Technologies Corporation and Sikorsky Aircraft. IMA was initially developed to aid in helicopter maintenance operations, but is now being sold as a general equipment maintenance tool.

A field mechanic uses the active document interface to communicate with the expert system by selecting equipment condition choices presented on each screen. The choice made is sent by the document to the expert system, which uses failure history and test data to determine the next test to be performed by the mechanic. After sending a choice to the expert system, the document then asks it how to construct the next page. The expert system passes back an identifier for the next equipment diagram (which also includes instructions and a question) and a small set of choices that the document presents in a set of active buttons on the bottom of the page.

This application was developed using existing systems. Not only did the expert system independently exist, but so did all the equipment diagrams. All that was needed for this application was a document framework to link these pieces together. An IMA document uses I6 process communication Lisp to communicate to the expert system, Lisp code to activate the choice buttons, and Lisp code to construct following screens based on expert system responses.

IMA is currently being adapted to operate on wearable computers, which include a head-mounted optical display and a voice activation interface. These will allow mechanics to work in cramped spaces with both hands free for equipment maintenance. Since the UI is a document, it is easily changed by adjusting fonts and pagination to accommodate smaller screens.

Document system facilities used by this application include printing, filtering to accept CALS [16] compliant documents, and automatic hypertext linking and indexing.

7 EXAMPLE: ORACLE COAUTHOR

CoAuthor is a self-learning proofreading tool used to check spelling, punctuation, and grammar. It can also check for simplified English [17], as well as look for common mistakes made by non-native English speakers.

The CoAuthor UI is implemented by a document beyond the lines described by Bier and Goodisman [7], as it has implemented a complete widget set using document objects. These include standard UI elements such as push buttons, toggle and radio buttons, text entry fields, and scrolling lists. See Figure 2.

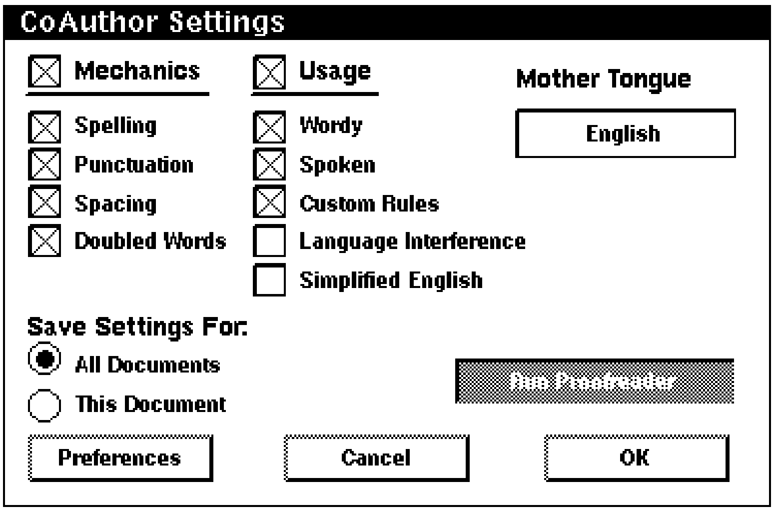


Figure 2. CoAuthor active document UI

The dialog windows are actually active documents that are tightly integrated with the user's source document being checked. A proposal for such an application was advanced by Terry and Baker [8].

To build this application, I6 Named Graphic Objects (NGOs) were constructed for each type of UI element. NGOs are objects created with standard I6 diagramming tools, then named by the user for easy reuse. In addition to NGOs giving end-users the ability to manage graphic objects by name, the document programmer can easily attach different methods on each type of NGO. Such a facility is essentially that provided by many stand-alone GUI builder systems.

When the user launches the CoAuthor dialog, it programmatically changes the appearance and content of its NGOs to show the current state of the proofreader. The user can then click, drag, or enter text into the NGOs to call the action specified for each NGO document object. These actions can modify the source document and/or the CoAuthor dialog. In addition, they can generate an interactive error report, which allows users to examine and deal with each error separately. Each error is linked to the source of the error in the document being checked.

In using document objects to represent UI elements, there are two common approaches used to change their appearances in I6 documents. First, one can programmatically apply different graphic properties or transformations to the object, such as changing its color or size or position. Second, one can use the conditional document assembly mechanism [18]

to hide one document object quickly (such as an empty box) and instead show another object (such as a filled box).

CoAuthor made extensive use of document objects to represent UI elements that are standard in many GUIs. This had advantages because these elements could be edited and placed using the standard I6 document editor, and thus treated the same as other content created by the document developer. However, while document objects can be useful in emulating standard GUI toolkits, if it is a goal to be GUI compliant it is simpler to use actual GUI widgets within the document.

Users do not know that they are interacting with documents when they use the CoAuthor dialogs—indeed, one of the points of this example is that active documents are useful not only in presenting new UI paradigms, but also for easily constructing systems that use existing paradigms. The IMA application described above is an example of the former—its UI paradigm is that of a paper document. CoAuthor is an example of the latter—it uses Motif-like UI paradigms, although constructed with document objects.

8 ACTIVE DOCUMENTS SOLUTIONS VERSUS OTHER SOLUTIONS

When comparing active document solutions to other solutions, there are two things to ask. First, is the UI document-centric? And second, is the application built from a document object system (an active document engine), or from some other toolkit?

All of the examples described above could have been implemented using different toolkits than an active document engine. However, if they were implemented with another tool, not starting from a document class library, and resulted in the same UI and functionality, in fact they would be active documents—but active documents that were harder to build than building them on a document system.

The problems solved by the tools described above could have been solved using different UI paradigms as well—they did not need to be solved as active documents.

The key advantages for active document applications are the following:

- Ease of use. As mentioned earlier, building an application that uses the document metaphor can make an application easily understood by a very wide audience.
- Ease of implementation. Document object systems have rich capabilities for creating interfaces. The application designer can use existing objects from a document object catalog, or s/he can easily create new objects using the document editor.
- Core publishing functionality. If an application has a need for text and graphics presentation and editing, document system based solutions are much easier to use and have more powerful functionality than is available in general application toolkits. A simple example is the ability to have mixed fonts and/or text properties in part of your UI. A more advanced example is the ability to do automatic pagination of floating graphic frames in a document with tables and images. Scenarios such as these are easy to do with document-based solutions, but are not available in most parts of standard application toolkits.
- External publishing functionality. In addition to having the presentation and editing functionality within the document, document system based applications can also take advantage of all the other publishing facilities that exist within the document system. For example, the IMA application makes use of the system's printing and filtering capabilities.

- Performance. Document systems are usually carefully optimized for high performance text and graphics. Complex user interfaces built from document objects can be more efficient than ones built from a client/server widget based system. For example, the active document can be constructed ahead of time and saved in its proprietary performance-optimized format. This allows faster *open* of such interfaces than those that have to do a great deal of client/server communication to build each widget.
- GUI portability. Depending on the paradigm used to construct the UI, document-based applications can be constructed to be portable across many GUI systems. Some of this is due to GUIs not having different standards for document-like text and graphics presentation. Most GUIs specify the *trimmings* more than the document presentation. Thus, if you take advantage of the document metaphor, your application has a good chance of working well across different GUI standards. Document-based applications that use native GUI objects have the same portability issues as generic application development kits, namely the tradeoff between minimal but portable interfaces compared to rich but non-portable ones. I6 allows either approach. One example of the first was used in an Interleaf 5 (I5) set of visual tools for the customization of the user's document processing environment. We refer to its simplified UI as *Point And Pound*, as it mostly consists of GUI independent buttons that the user points at and selects to trigger activity. By contrast, I6 also allows the use of native widgets.

9 IMPLEMENTATION NOTES

9.1 What should an active document enabling system contain?

The most important element in an active document system is the underlying structured document model. It should contain a rich set of document objects (text, components, tables, autonumbers, page numbers, index tokens, reference objects, footnotes, frames, diagrams, beziers, images, files, documents, books, etc.), and a rich set of properties for each object.

There must be programmatic access to do everything that is possible from the UI, and more. There should be access for object creation and destruction; object navigation (ideally through different views, such as format and structure); getting and setting predefined and user-defined properties.

The system should be object-oriented to allow for easy active document building and reusability of active objects.

There should be a set of editor objects, which can act as user agents to intercede on behalf of the user between the UI and the underlying document objects. Editor objects are responsible for side effects such as posting choice dialogs and reporting error messages.

The system should have a full development environment with online hypertext documentation, active document development tools, integration with standard development tools, and support.

9.2 Active document seeding

An interesting aspect of active document development is how objects get *seeded*—that is, how otherwise static objects first get activity attached to them. In I6 the developer first evaluates some code to get a handle on an object. This can be done programmatically by

navigating from the current document object, or by asking for the object at the cursor. Then the developer typically evaluates code to change the class of that object and to provide new methods for that class.

An object can be under edit by a normal document editing session, as well as under programmatic edit. Developers of active documents under I6 use both methods, the former often being easier. For example, a developer usually enters text or graphics into the active document by using the built-in editor for such objects. But the developer also often evaluates his or her document Lisp scripts to test them on the active document under edit. There is no race condition in permitting both types of editing, since there is only one event loop in the system. I6 processes end-user editing requests and programmatic requests in the order in which they arrive, whether from the current or an external process. The only difficulty of having documents open for both interactive and programmatic editing arises when some lower level programmatic interfaces mistakenly do not queue up redisplay requests, which leads to temporarily confused displays.

Once an object is made active, the developer has to provide a way for that object to remain active even after its document has been saved and closed. This can be done by providing an object *save* method. The document sends *save* messages to all objects at document save time, passing them the actual file stream object. A custom save method can write out code to be evaluated at open time, thereby reactivating such objects. The custom *save* method is usually a *before* or *after* method, which gets called *in addition to* the default save method for that object type. The default save method simply writes out the object itself.

An alternative to providing an object save method (which saves object-specific code and/or data next to the object in the main document file stream) is to save code on the document itself, either within the main document file stream or in an auxiliary file containing method definitions. The current version of I6 allows a document to be represented in several different *part files*, to allow high performance updating for some data such as autonumbers, and to allow some updates without needing to lock the main document file against editing. One such part file is the *methods* file, described below.

9.3 Activation; security issues

Active behavior occurs in these ways:

- If a document has an auxiliary methods file, it is loaded and evaluated when that document is first encountered by the software. Many I6 active documents only define their classes and autoload their methods in initialization contained in the methods file.
- If a document contains active objects within the document file stream (as opposed to the methods file), these become activated when the document is opened programmatically or by the user for viewing, editing, or printing.
- At software startup, all objects within the user's *profile* container are sent a *load* message, calling them to action. Users thus place initialization documents or scripts in their profile container.
- Document objects can have user-defined attributes for many purposes, such as to mark conditional content or to store data to be accessed by Lisp programs. There are some specially named attributes which, when they exist, are evaluated at document open

time. For example, such attributes are used by office memo templates to initialize the content of the *Date* and *From* fields.
- Users can *explicitly* activate an object by selecting it and issuing a Load menu command, or by explicitly evaluating some code that does this.

Users are not always cognizant of the first four scenarios above. Someone could simply copy an active document into a user directory, and when that user opened the directory, the activity would be triggered, possibly with undesirable results.

The system contains two facilities to limit activation. The first is to run the software with a *-static* option on the command line, which disables automatic activation described in the first four scenarios listed above. The second is to define an "active load hook" that gets called prior to each automatic activation. It is passed the current object and the code that is about to be evaluated. A sophisticated program can examine this code and decide whether or not to permit its activation. We note that examination of code by code is easily done in Lisp, which makes no distinction between code and data. A simple example of such an inspector is distributed with the system.

9.4 Active document delivery issues

I6 active documents can be completely self-contained and easily activated. Simply copying such a document from one filesystem to another *installs* that active document application. This suffices since the object methods are stored within the document, and the copy of the document will have its active objects activated exactly as the original.

The active document implementor can choose between building such self-contained documents (which contain all their own data and code) and building "layered applications." The latter use a code library scheme, in which the code is installed in a standard place, to be shared by all instances of that document class. Self-contained active documents are easier to move around, but in cases where there are many instances, a library scheme saves space and makes updates easier. Interleaf has delivered some self-contained active document applications that can automatically update themselves if a newer version has been encountered.

In our judgement, one important requirement for automatic activation of documents is run-time binding in the document extension language—this is one of the reasons we chose Lisp. With languages such as C++, for example, newly introduced active documents or active objects must first be compiled into existing systems.

10 CONCLUSIONS

Our experience with active documents leads us to the following positions:

- Lisp is a good language for active document development due to features such as the interpreter (which allows nice development tools), functions as data (it is easy to store and manipulate programs as data on document objects), and run-time binding (which is important for the delivery of active document objects).
- Use of the object system simplifies object activation, as discussed in the multimedia photo album example.
- Active documents are useful for both document-based UI (e.g., IMA) and emulation of standard UI paradigms (e.g., CoAuthor).

- Active documents need not appear active to all users (e.g., the auto-localizing templates).
- It is difficult to catalog complete requirements for an active document building system without having many examples. I6 extensibility benefited from experience with several dozens of active document applications built with I5.

We believe the following areas need more research:

- Non-programmer tools for building active documents.
- UI guidelines for active documents, such as how to switch between build and run modes.
- UI experimentation on how the document UI model fits into standard GUI environments —mixing document objects and standard GUI widgets.
- Tools for finding, editing, and debugging active objects within a document.

ACKNOWLEDGEMENTS

Thanks especially to Ed Blachman and Robert A. Morris for participation in a mini-seminar on active documents, and for careful review of this paper. Thanks also to several of our Interleaf colleagues for comments. Thanks to Francis H. Yu of Oracle Corporation and to Laryssa Kachorowsky and John Wronski of Sikorsky Aircraft for information about their active documents.

USAGE INFORMATION

Interleaf 6 (I6) is being released in October 1993. Interleaf 5 (I5) was released in 1989, and currently has over 200 000 users. The Developer's ToolKit (DTK) is a set of development tools and documentation used to customize I5, integrate it with other applications, and to build active documents and document-based applications. Over 1000 copies of the DTK have been sold, and Interleaf's customer support organization has worked closely with over 100 DTK customers. Internally more than 100 employees (in field offices, in engineering, documentation, QA, marketing, and customer support) have written some type of application based on I5. Universities can generally receive full I5 licenses (including DTK) for only the cost of media. We have not tracked university use of our DTK, although we have seen discussions and examples posted on Usenet.

REFERENCES

1. Bertrand Meyer, *Object-oriented Software Construction*, Prentice Hall, Hemel Hempstead, Herts., England, and Englewood Cliffs, N.J, U.S.A., 1988.
2. Guy L. Steele Jr., *Common LISP, The Language*, Digital Press, 1990.
3. Richard M. Stallman, 'Emacs, the extensible, customizable self-documenting display editor', *SIGPLAN Notices*, **16**(6), 147–56, (June 1981).
4. Robert Spinrad, 'Dynamic documents', *Harvard University Information Technology Quarterly*, **VII**(1), 15–18, (Spring 1988).
5. Polle T. Zellweger, 'Active paths through multimedia documents', in *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography* [19], pp. 19–34.

6. Dennis Arnon, Richard Beach, Kevin McIsaac, and Carl Waldspurger, 'Camino real: an interactive mathematical notebook', in *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography* [19], pp. 1–18.
7. Eric A. Bier and Aaron Goodisman, 'Documents as user interfaces', in *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography* [20], pp. 249–262.
8. Douglas B. Terry and Donald G. Baker, 'Active Tioga documents, an exploration of two paradigms', in *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography* [20], pp. 105–122.
9. Daniel Swinehart, Polle T. Zellweger, Richard Beach, and Robert Hagmann, 'A structural view of the Cedar programming environment', *ACM Transactions on Programming Languages and Systems*, **8**(4), 419–490, (1986).
10. Donald D. Chamberlin, Helmut F. Hasselmeier, and Dieter P. Paris, 'Defining document styles for WYSIWYG processing', in *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography* [19], pp. 121–138.
11. George Towner, 'Auto-updating as a technical documentation tool', in *Proceedings of the 1988 ACM Conference on Document Processing Systems* [21], pp. 31–36.
12. Wilfred J. Hansen, 'Enhancing documents with embedded programs: how Ness extends insets in the Andrew ToolKit', in *Proceedings of the 1990 IEEE International Conference on Computer Languages*, pp. 23–32, New Orleans, (12–15 March 1990).
13. Yaron Goldberg, Marilyn Safran, and Ehud Shapiro, 'Active mail—a framework for implementing groupware', in *Proceedings of the ACM 1992 Conference on Computer-Supported Cooperative Work*, pp. 75–83. ACM Press, (1992).
14. Paul M. English, Ethan S. Jacobson, Robert A. Morris, Kimbo B. Mundy, Stephen D. Pelletier, Thomas A. Polucci, and H. David Scarbro, 'An extensible, object-oriented system for active documents', in *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography* [20], pp. 263–276.
15. Ramana Rao, Stuart K. Card, Herbert D. Jellinek, Jock D. Mackinlay, and George G. Robertson, 'The information grid: a framework for information retrieval and retrieval-centered applications', in *Proceedings of the Fifth Annual ACM Symposium on User Interface Software and Technology*, pp. 23–32. ACM Press, (1992).
16. 'Markup requirements and generic style specification for electronic printed output and exchange of text', Technical Report Military Specification MIL-D-28000, CALS Policy Office, DASD(S) CALS, Pentagon, Room 2B322, Washington, D.C., (1988).
17. 'Specification for manufacturers' technical data', Technical Report A.T.A. Specification No. 100, Revision No. 30, AIR Transport Association of America, 1709 New York Avenue N.W., Washington, D.C. 20006, (1991).
18. Richard Ilson, 'Interactive effectivity control: Design and applications', in *Proceedings of the 1988 ACM Conference on Document Processing Systems* [21], pp. 85–92.
19. *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography*, Nice (France), April 20–22 1988. Cambridge University Press.
20. *Proceedings of the International Conference on Electronic Publishing, Document Manipulation and Typography*. Cambridge University Press, September 1990.
21. *Proceedings of the 1988 ACM Conference on Document Processing Systems*, Santa Fe, New Mexico, 5–9 December 1988. ACM Press.

US005579519A

United States Patent [19]

Pelletier

[11] **Patent Number: 5,579,519**

[45] **Date of Patent: Nov. 26, 1996**

[54] **EXTENSIBLE ELECTRONIC DOCUMENT PROCESSING SYSTEM FOR CREATING NEW CLASSES OF ACTIVE DOCUMENTS**

[75] Inventor: **Stephen Pelletier**, San Francisco, Calif.

[73] Assignee: **Interleaf, Inc.**

[21] Appl. No.: **923,937**

[22] PCT Filed: **Mar. 4, 1991**

[86] PCT No.: **PCT/US91/01479**

§ 371 Date: **Feb. 16, 1994**

§ 102(e) Date: **Feb. 16, 1994**

Related U.S. Application Data

[63] Continuation-in-part of Ser. No. 489,176, Mar. 5, 1990, abandoned.

[51] **Int. Cl.**[6] **G06F 9/44**

[52] **U.S. Cl.** **395/705**; 395/776; 364/DIG. 1.61; 364/225.6; 364/250; 364/280.4

[58] **Field of Search** 395/700, 650, 395/600

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,633,430 12/1986 Cooper 395/144
4,996,662 2/1991 Cooper et al. 395/600
5,307,499 4/1994 Yin 395/700
5,333,237 7/1994 Stefanopoulos 395/12

OTHER PUBLICATIONS

Sobell, Mark G., "A Practical Guide to the Unix System", Benjamin/Cummings, 1989, pp. 66–67.

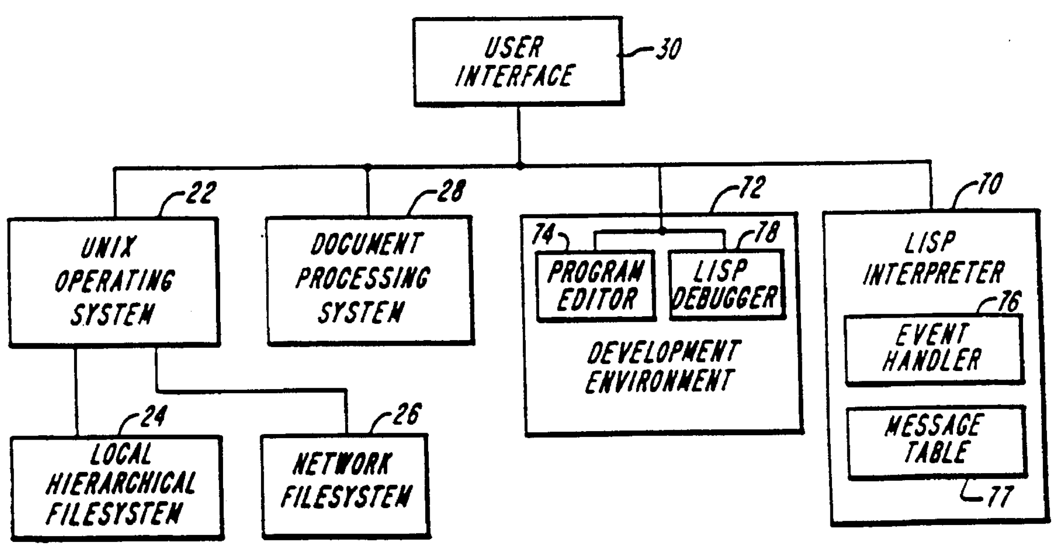
Hares, John S. and Smart, John D., "Object Orientation", John Wiley & Sons, 1993, pp. 42–44.

Primary Examiner—Kevin A. Kriess
Attorney, Agent, or Firm—Hale and Dorr

[57] **ABSTRACT**

An improved document processing system for creating computer procedures (i.e., methods) and associating them with an electronic document structured as a nested hierarchy of constituent objects. The system includes a document processing means for creating the electronic document, and a program creating means for defining the method and associating it with the electronic document or its lower level constituent objects. The system further includes an interpreter for interpreting the computer procedure in response to a specified event.

20 Claims, 6 Drawing Sheets



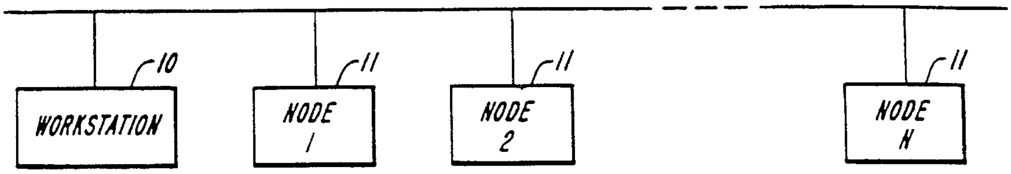


FIG. 1B

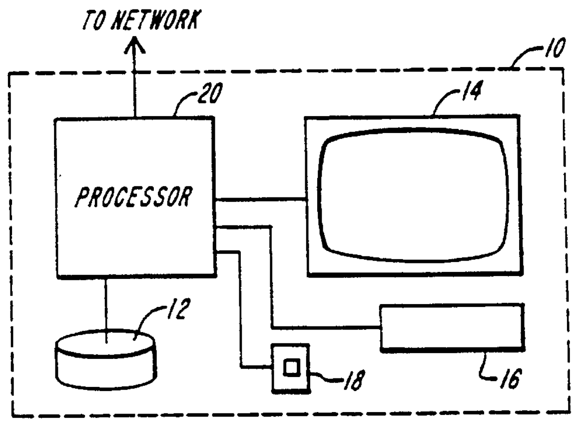


FIG. 1A

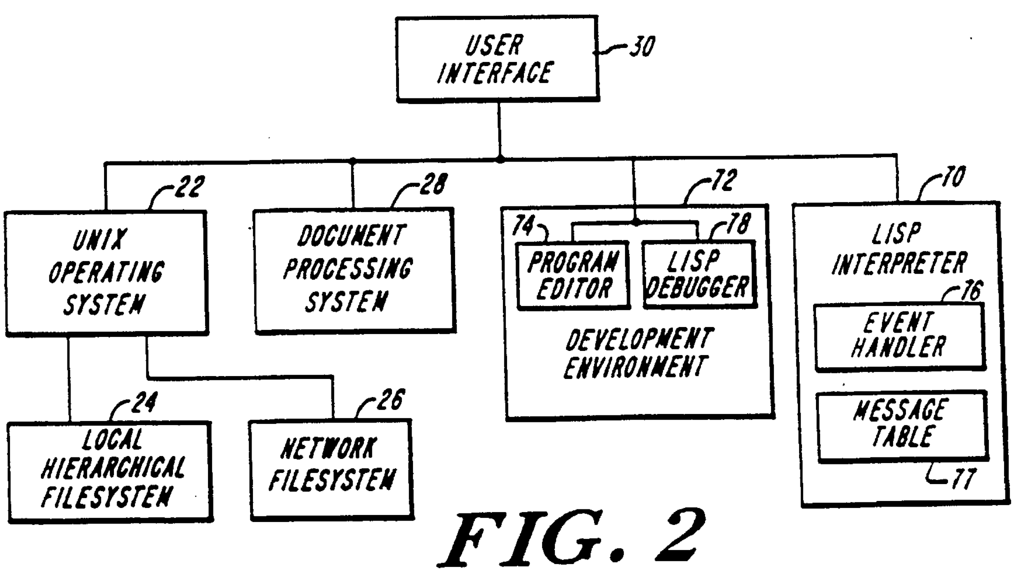


FIG. 2

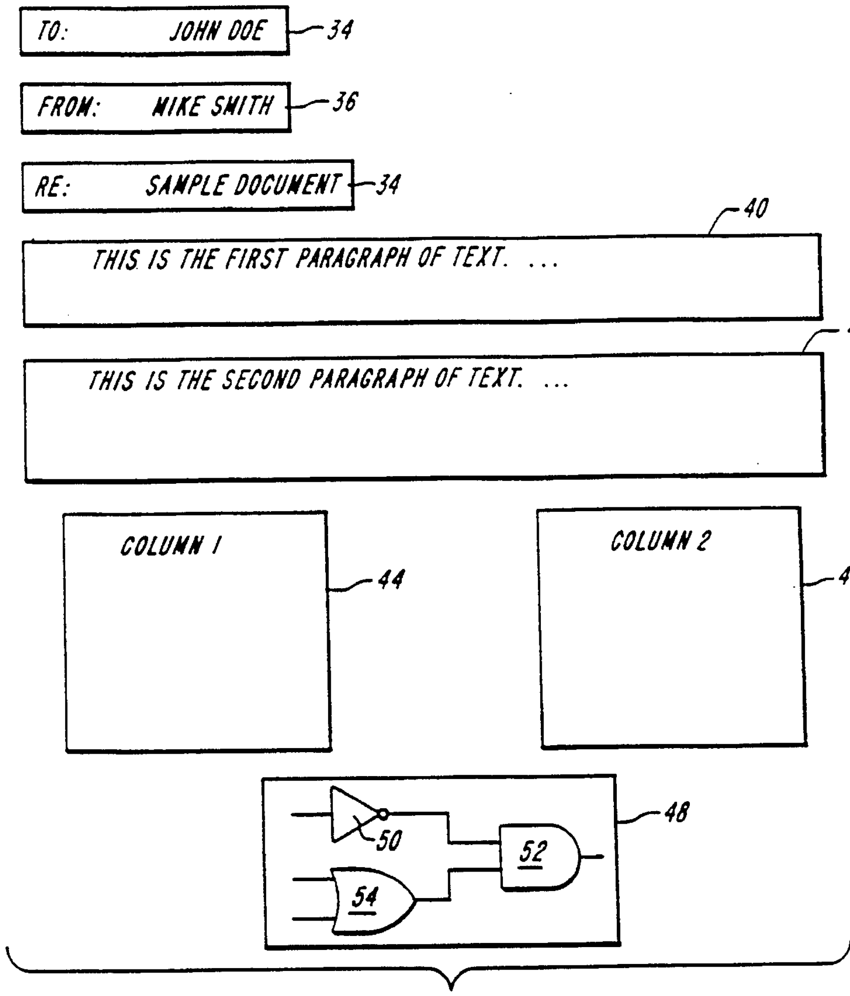

TO: JOHN DOE
34
FROM: MIKE SMITH
36
RE: SAMPLE DOCUMENT
34
40
THIS IS THE FIRST PARAGRAPH OF TEXT. ...
42
THIS IS THE SECOND PARAGRAPH OF TEXT. ...
COLUMN 1
44
COLUMN 2
46
48
50
52
54

FIG. 3

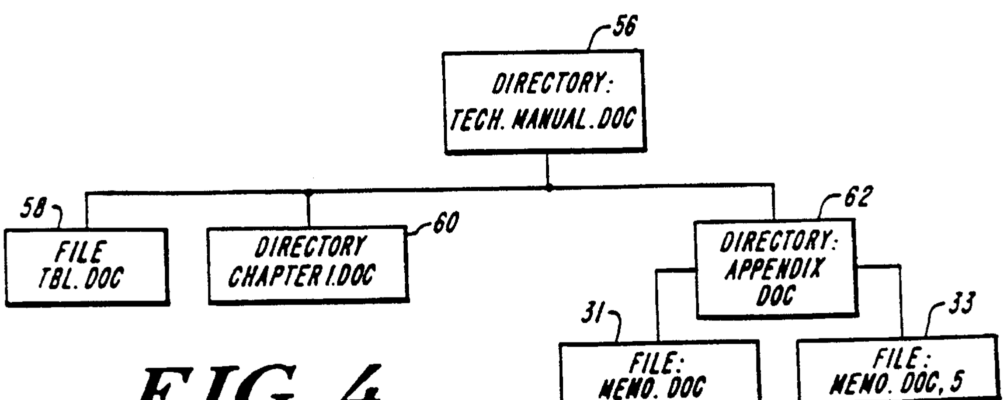

56
DIRECTORY:
TECH. MANUAL.DOC
58
FILE
TBL.DOC
DIRECTORY
CHAPTER 1.DOC
60
62
DIRECTORY:
APPENDIX
DOC
31
FILE:
MEMO.DOC
33
FILE:
MEMO.DOC,5

FIG. 4

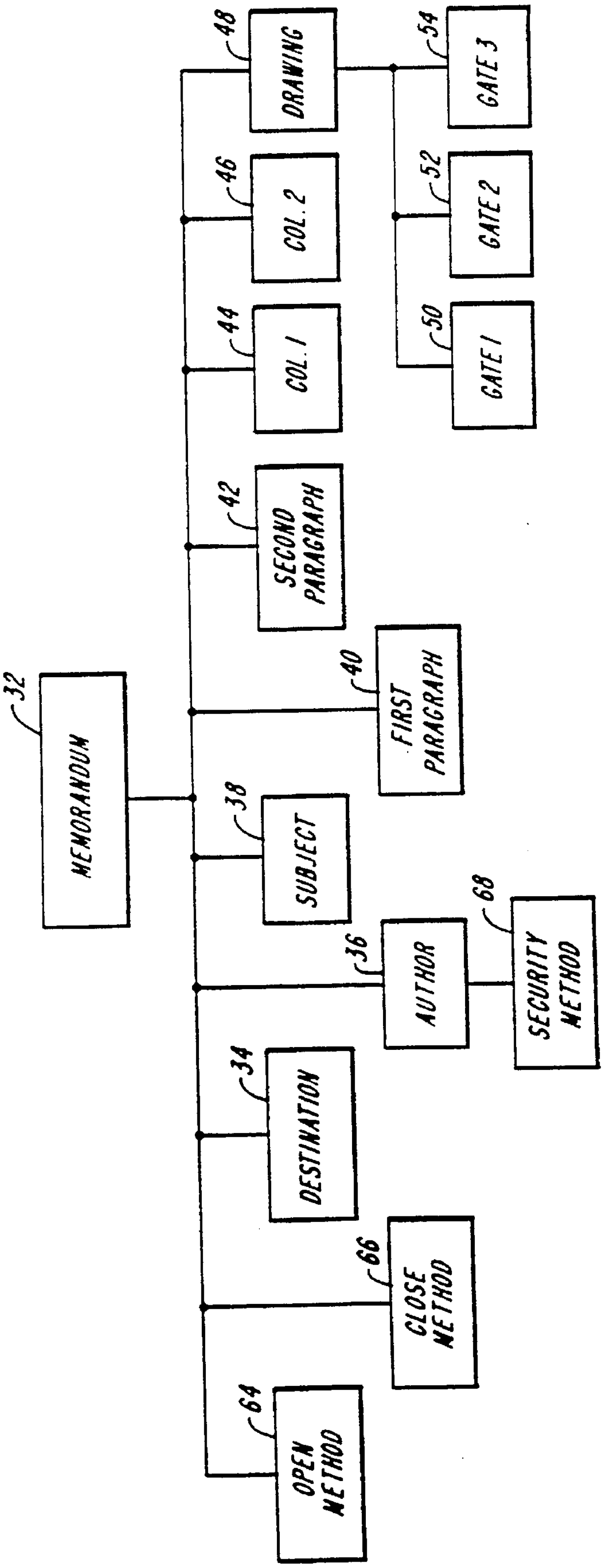

32
MEMORANDUM
34
DESTINATION
36
AUTHOR
68
SECURITY METHOD
38
SUBJECT
40
FIRST PARAGRAPH
42
SECOND PARAGRAPH
44
COL. 1
46
COL. 2
48
DRAWING
50
GATE 1
52
GATE 2
54
GATE 3
64
OPEN METHOD
66
CLOSE METHOD

FIG. 5

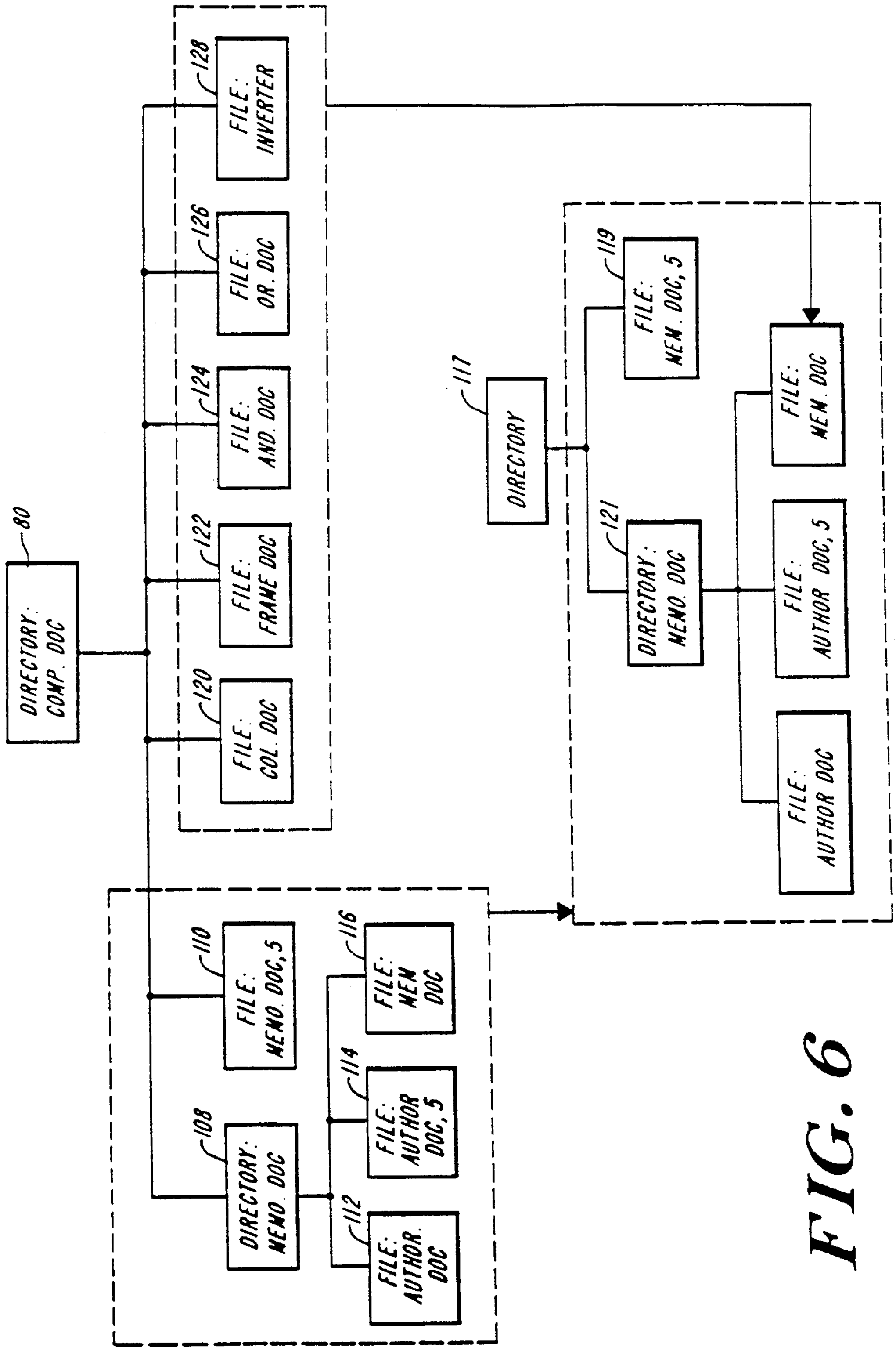

80
DIRECTORY: COMP. DOC
120
FILE: COL. DOC
122
FILE: FRAME DOC
124
FILE: AND. DOC
126
FILE: OR. DOC
128
FILE: INVERTER
108
DIRECTORY: MEMO. DOC
110
FILE: MEMO. DOC, 5
112
FILE: AUTHOR DOC
114
FILE: AUTHOR DOC, 5
116
FILE: MEM DOC
117
DIRECTORY
119
FILE: MEM. DOC, 5
121
DIRECTORY: MEMO. DOC
FILE: MEM. DOC
FILE: AUTHOR DOC, 5
FILE: AUTHOR DOC

FIG. 6

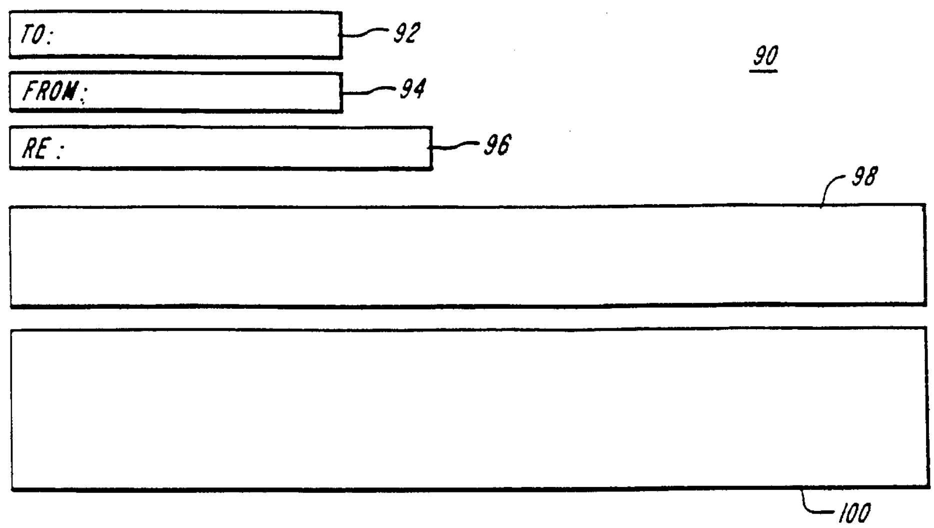

TO:
92
FROM:
94
RE:
96
90
98
100

FIG. 7

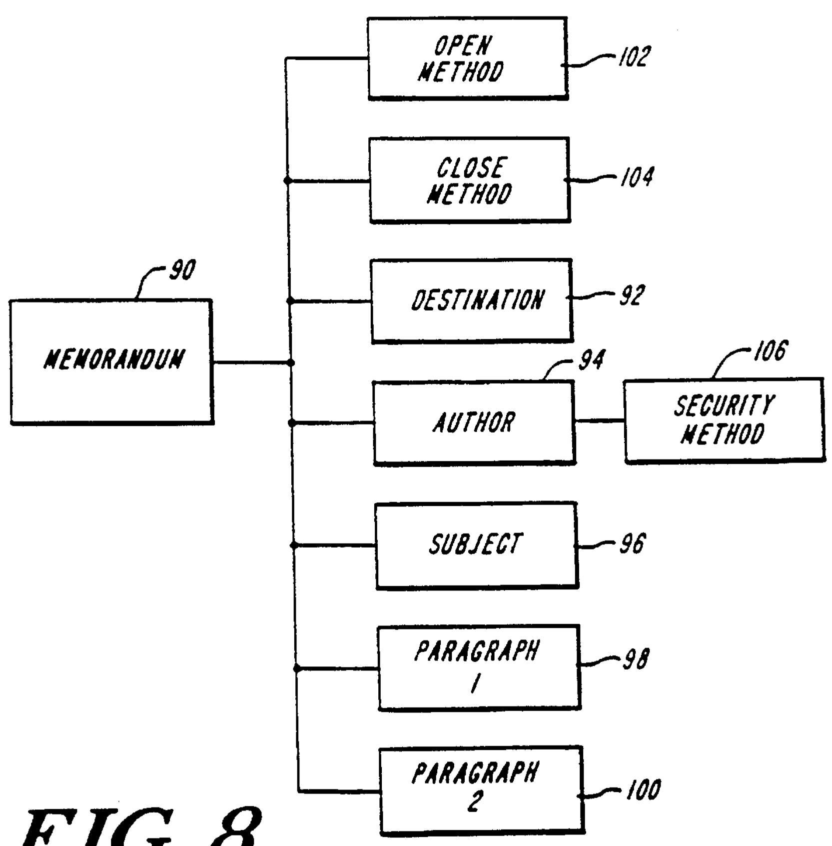

OPEN METHOD
102
CLOSE METHOD
104
DESTINATION
92
90
MEMORANDUM
94
AUTHOR
106
SECURITY METHOD
SUBJECT
96
PARAGRAPH 1
98
PARAGRAPH 2
100

FIG. 8

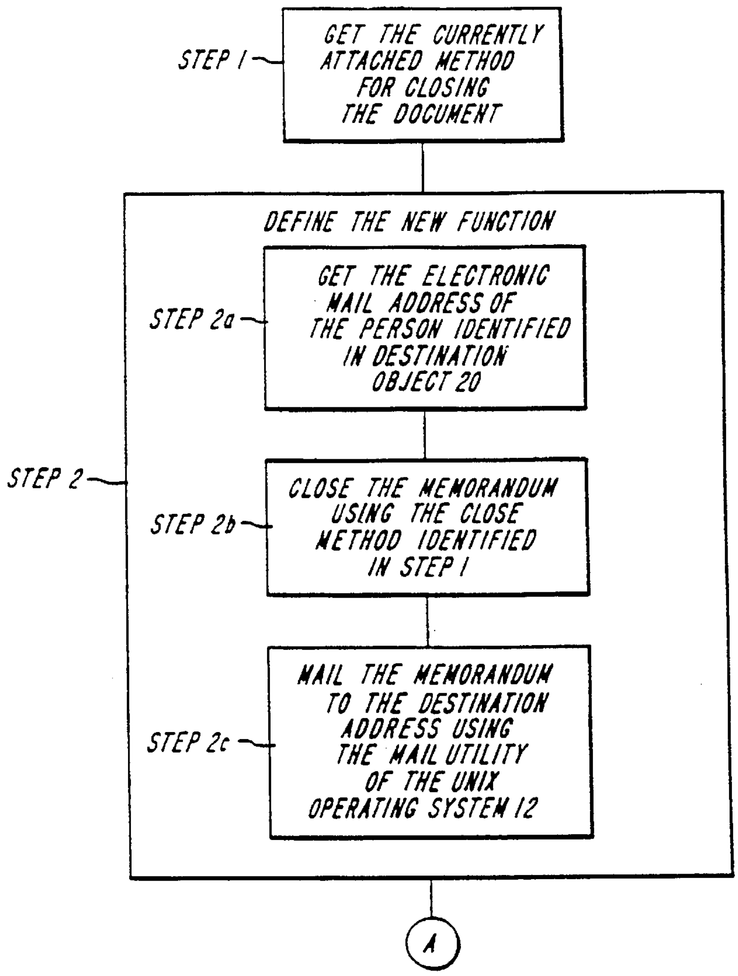

STEP 1
GET THE CURRENTLY ATTACHED METHOD FOR CLOSING THE DOCUMENT
STEP 2
DEFINE THE NEW FUNCTION
STEP 2a
GET THE ELECTRONIC MAIL ADDRESS OF THE PERSON IDENTIFIED IN DESTINATION OBJECT 20
STEP 2b
CLOSE THE MEMORANDUM USING THE CLOSE METHOD IDENTIFIED IN STEP 1
STEP 2c
MAIL THE MEMORANDUM TO THE DESTINATION ADDRESS USING THE MAIL UTILITY OF THE UNIX OPERATING SYSTEM 12
A

FIG. 9A

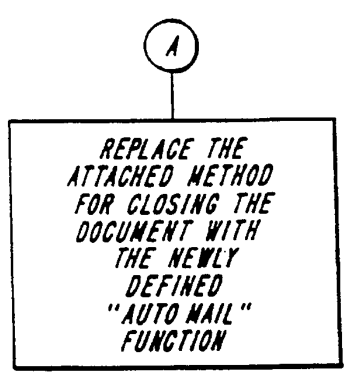

A
REPLACE THE ATTACHED METHOD FOR CLOSING THE DOCUMENT WITH THE NEWLY DEFINED "AUTO MAIL" FUNCTION

FIG. 9B

EXTENSIBLE ELECTRONIC DOCUMENT PROCESSING SYSTEM FOR CREATING NEW CLASSES OF ACTIVE DOCUMENTS

This application is a national stage application of PCT/US91/01479, which is a continuation-in-part of Ser. No. 07/489,176 filed Mar. 5, 1990, now abandoned.

This invention relates to a Document Processing System for creating, editing, printing and presenting active electronic documents which are associated with computer programs.

BACKGROUND OF THE INVENTION

Electronic Document Processing Systems generally include a computer workstation programmed to allow a user to create and edit an electronic representation of a document. The workstation includes a display device for displaying an image of the document as it would appear if printed. For example, a document such as a magazine article may contain an arrangement of text and graphics. The system allows the user to edit the document by modifying existing text and graphics or by adding entirely new text or images. As the user modifies the document, the display is continually updated to reflect these changes, thereby allowing the user to interact with the system to achieve the desired document.

SUMMARY OF THE INVENTION

We have discovered an improved document processing system for creating computer procedures (i.e,. methods) and associating them with an electronic document structured as a nested hierarchy of constituent objects. The system includes a document processing means for defining the method and associating it with the electronic document or its lower level constituent objects. The system further includes an interpreter for interpreting the computer procedure in response to a specified event.

In preferred embodiments, the methods are defined in a dialect of the programming language LISP. A LISP interpreter interprets the new LISP programs, thereby allowing the user to add new methods without modifying the underlying document processing system.

DESCRIPTION OF THE DRAWINGS

FIG. **1**(*a*) is a block diagram of a computer workstation programmed to operate as a document processing system.

FIG. **1**(*b*) is a block diagram illustrating a computer network which includes the computer workstation shown in FIG. **1**(*a*).

FIG. **2** is a block diagram of the software architecture of an electronic document processing system embodying the present invention.

FIG. **3** is an illustration of an electronic document as it would appear if displayed or printed.

FIG. **4** is a diagram of a directory, within the Unix filesystem which contains a high level document.

FIG. **5** is a schematic of the hierarchy of data objects and methods comprising the memorandum document shown in FIG. **3**.

FIG. **6** is a diagram of a directory within the Unix filesystem which contains generic document objects.

FIG. **7** is an illustration of a generic memorandum document as it would appear if printed.

FIG. **8** is a schematic of the hierarchy of generic data objects and methods comprising the generic memorandum object shown in FIG. **7**.

FIGS. **9**(*a*) and (*b*) is a flow chart of a LISP program for defining a new method and attaching it to a document.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to FIG. **1** (*a*) an electronic document processing system **10** (herein "EDPS" **10**) is implemented using a properly programmed computer workstation. The workstation includes a disk **12** for storing electronic representations of documents, and a display **14** for displaying documents for viewing by a user. The workstation is connected to an Ethernet network, thereby allowing the workstation to access documents and software stored on other nodes **11**. (FIG. **1**(*b*))

The user selects a document by manipulating a keyboard **16** and/or a mouse **18**. A processor **20**, in response to the user's input, retrieves an electronic representation of the selected document from disk **12** and supplies display **14** with image data representative of the selected document. In response, display **14** generates an image of the document for viewing by the user.

The user interacts with the system to modify the selected document. For example, the user may enter keyboard commands requesting the system to select certain text and delete it from the document. Processor **20** responds to the commands by editing the electronic representation of the document and modifying the image data to adjust the displayed image.

In addition to modifying the appearance of a document, a user may define a procedure to be associated with the document. The associated procedure is performed upon the occurrence of a specific event. For example, a procedure may be defined which electronically mails the document to another user as a reminder that a deadline is approaching. When the system clock indicates that a certain date has arrived, the procedure automatically mails the document. Documents having such associated procedures are referred to herein as "active" documents.

Referring to FIG. **2**, the software for implementing the document processing system is now described in more detail. EDPS **10** includes a Unix Operation System **22**. The Operating System maintains on disk **12** a local hierarchical filesystem **24** which contains documents and their associated methods. It also provides access to a network filesystem **26** which includes files from remote nodes **11** which contain documents or software used by EDPS **10**.

The system also includes a "WYSIWYG" document processing software **28** (herein "DPS" **28**) (Note: "WYSIWYG" is a common acronym for "what you see is what you get". Such systems allow the user to view the document as it would appear if printed and interact with the system to modify the document image.) DPS **28** is an object oriented document processing system which utilizes the Unix Operating System **22** and User Interface **30** to create and modify documents.

Documents may be any type of written instrument (e.g., a memorandum or a technical manual). DPS **28** structures documents as a nested hierarchy of data objects. For example, referring to FIGS. **3** and **5**, a sample memorandum **32** includes an author object **34** identifying the author of the memorandum, a destination object **36** identifying the person to whom the memorandum is addressed, and a topic object

38 summarizing the subject of the memorandum. Memorandum **32** further includes two paragraph objects **40** and **42** which contain the text of the memorandum, a pair of column objects **44**, **46** each containing a list, and a graphic object **48** containing a line drawing. Each object **34–48** may consist of a plurality of lower level objects. For example, each word or character of paragraph **40** may be treated as a separate object and each symbol representing a logic gate **50–54** of drawing **48** may be treated as a distinct object. Further, the hierarchy may extend upward from the memorandum. For example, the memorandum may be a low level object attached to an appendix which in turn is connected to higher level super document (e.g., a technical manual).

The electronic representation of the document is stored by DPS **28** in one or more files within the local hierarchical filesystem **24**. Referring to FIG. **4**, for example, DPS **28** may utilize Operating System **22** to create a directory **56** which includes subdirectories and files containing all objects of a technical manual. Directory **56** may include a file **58** containing an electronic representation of a table of contents. Further, it may include a subdirectory **60** which may include an entire tree (not shown) of files and subdirectories containing objects which comprise the first chapter of the manual.

Thus, DPS **28** uses the file structure of the Unix Operating System to provide a hierarchical structure of data objects. Yet even within a single file of the Unix filesystem, DPS **28** may add additional layers of objects. For example, in the file structure shown in FIG. **4**, the electronic representations of all data objects **34–48** of the memorandum document **32** are stored in a single file **31** within an Appendix directory **62**. The data objects **34–48** within memorandum file **31** include at least two layers of objects, e.g., the symbols representing logic gates **50–54** are children of the parent graphic object **48**.

As explained above, any data object within the hierarchy may be associated with a method which, upon the occurrence of a certain event, will be implemented. For example, referring to FIG. **5**, memorandum **32** is associated with an open method **64**. When a user, via user interface **30**, indicates that he desires to view memorandum **32**, the open method **64** is implemented to open the memorandum file **31** and prepare image data representative of how the document would appear if printed. The image data is supplied to display device **24** which generates an image of the document as shown in FIG. **3**. Similarly, when the user indicates the desire to close memorandum **32**, close method **66** is implemented to store the electronic representation of the memorandum and remove its image from the display.

Methods may be attached to an object anywhere within the nested hierarchy, e.g., to a technical manual which is near the top of the hierarchy, to a memorandum, or even to a relatively low level object such as author object **36** (see e.g., security procedure **68**).

A method is a collection of instructions specified in a dialect of the language LISP. The collection of LISP instructions which define the method are stored in a file within the local hierarchical filesystem **24**. For example, referring to FIG. **4**, open method **64** is defined by a set of LISP instructions stored in file **33**. To indicate that the open method is associated with memorandum **32**, DPS **28** stores the method file **33** in the same directory as the memorandum file **31** and assigns the method file **33** a Unix filename comprising the filename of memorandum file **31** with the suffix ",5". For example, if memorandum file **31** is named "memo.doc", file **33** is assigned the name "memo.doc,5".

Several active objects may have the same hierarchical structure of data objects and attached methods. They differ only in the content of their data objects. Such objects are considered to be instances of the same object class. For example, paragraph objects of the same class have the same procedures for setting margin and font. However, the text of each paragraph object is unique to the paragraph object.

To facilitate the creation of active documents, the system includes a collection of predefined classes of active objects ranging in complexity from the lowest level object to high level objects at the top of the nested hierarchy. More specifically, for each predefined class, the system in effect includes a generic object which is an instance of the class. The generic object includes a hierarchy of empty data objects with attached predefined methods. A user may build a desired active document by copying selected generic objects (with their associated methods) and filling the empty data objects with the desired data.

The following illustrates this technique by describing the construction of a memorandum from several generic objects. Referring to FIG. **6**, local filesystem **24** includes a directory **80** which contains predefined generic document component objects which may be used as building blocks in constructing a desired document. For example, directory **80** may include a boilerplate memorandum document object **90**. The image of the generic memorandum object **90** is depicted in FIG. **7**. It includes a destination object **92** which lacks data identifying the destination. Similarly, it includes an author object **94**, a subject object **96** and a pair of paragraph objects **98**, **100**, each having no data. The schematic of the generic document, shown in FIG. **8**, indicates that object **90** is associated with a pair of methods **102**, **104** for opening and closing the document. Further, the author object is associated with a security method **106**.

FIG. **6** illustrates the Unix files which contain the generic memorandum object **90** and its associated methods. Subdirectory **108** contains all of data objects within memorandum **90**. The methods **102**, **104** associated with object **90** are stored in file **110**. To indicate that the methods are associated with the memorandum object, directory **108** is named "memo.doc" and the file **110** containing the methods is named "memo.doc,5". The author object **94** is stored in a file **112** within subdirectory **108**. Similarly, its associated security method is stored in file **114** in the same directory. Finally, all remaining objects are stored in file **116** within subdirectory **108**.

DPS **28** copies the generic object by copying directory **108**, including all children, to a desired location, e.g., destination directory **117** (FIG. **4**). Similarly, it automatically copies its associated method file **110** to the same location. The user can then open each of the data objects within the new document to add data (e.g., to add a name to the destination object).

Assume the user wants to build a memorandum document such as memorandum **32**. Document object **90** lacks a pair of column objects and a graphic object similar to those of memorandum **32** (FIG. **3**). Accordingly, the user copies generic objects corresponding to the missing items from directory **80**. For example, file **120** contains an empty column object, file **122** contains an empty frame object for graphics, and files **124–128** contain graphic components such as components **50–54** of memorandum **32**. In response to the user's selection, DPS **28** appends copies of these generic objects to file (**121**) (See FIG. **6**), thereby constructing the desired memorandum. The system thus allows a user to construct a variety of new active documents by merging selected generic objects.

Referring to FIG. **1**, the system includes an interpreter **70** for interpreting the LISP programs. The interpreter includes a method table **77** for each class of documents which identifies the location of all LISP methods associated with the class and its lower level objects.

Generally, an object's method may be invoked in two ways, (1) in response to a LISP instruction within another method, or (2) in response to an external event directed to the object.

With regard to the first example, the LISP dialect provided by LISP interpreter **70** includes a "tell" instruction which includes two arguments, a Path Name identifying a specific object, and a Message Identifier identifying a "message" to be sent to the specified object. In response to a LISP message directed to a specified object, interpreter **70** searches the method table **77** to locate the method attached to the specified object which is responsive to the message. It then responds to the message by evaluating the method. For example, a LISP "tell" instruction may send an "open" message to memorandum **32**. In response, interpreter **70** locates and evaluates the memorandum's open method.

A method may also be triggered in response to an external event, e.g., an instruction from user interface **30** to open a specified document. However, user interface **30**, Unix Operating System **22** and other software which may generate such external events, are typically written in the Language "C". Accordingly, the LISP interpreter includes as event handler **76** which allows non-LISP programs to access methods written in the LISP language. More specifically, event handler **76** responds to a "C" language call instruction from software such as user interface **30** and Unix Operating System **22** by locating and evaluating the appropriate LISP code. For example, in response to a "C" language call from user interface **30** to open memorandum **32**, event handler locates method **64** and evaluates it.

The user may desire to create an active document having an associated method which is not available among the predefined generic classes of objects. Thus, the system further includes development environment **72** for allowing the user to create a new class of active objects by creating new methods and attaching them to an existing object.

To create a new method, the user, via user interface **30**, first invokes development environment **72**. Environment **72** allows the user to select, from all existing objects, an active object whose associated methods most closely match those desired in the new class. For example, suppose a user wants the above created memorandum object **90** to electronically mail itself whenever it is closed. With the exception of the "automail" method, the memorandum object **90** already has all the methods desired in the new class (e.g., open and close methods **64**, **66**). Accordingly, the user simply needs to draft the new method and attach it to the memorandum object.

Toward this end, environment **72** includes a program editor **74** for allowing a user to define the new method in the interpretive language LISP. (See FIG. **2**) For example, referring to FIGS. **9**(*a*), **9**(*b*) the user drafts a LISP program which first determines the identity of close method **104** which is currently attached to the copied object (step 1). The LISP program next defines a new function "automail" having the following three steps:

1. Get the electronic mail address of the person identified in destination object **92** (step 2a);
2. Close the memorandum using the close method **104** identified in step 1 above (step 2b), and
3. Mail the memorandum to the destination address using the mail utility of Unix Operating System **22** (step 2c).

After defining the new function, the next LISP instruction updates the method table **77** to replace the close method **104** currently attached to the memorandum with the new "automail" function defined above (step 3). Accordingly, evaluation of step 3 of the above LISP program attaches the "automail" function to the document by replacing close method **104** with the newly defined automail function.

To indicate that a new method has been added to memorandum **90**, development environment **72** stores the above LISP code in the method file **119**. Method file **119** is in the same directory **117** as subdirectory **121** which contains the data objects of memorandum **90**. Subdirectory **121** is named "memo.doc." and the method file **119** is named "memo.doc, 5". When DPS **28** next attempts to access the newly created memorandum, interpreter **70** will look to the method file associated with the memorandum, (i.e. "memo.doc.,5") and interpret the new LISP code, thereby updating the message table to include the newly defined method. Development Environment **72** further includes a LISP Debugger **78** which allows the user to test newly drafted LISP Code and locate errors in the code.

Interpreter **70** allows the user to create new methods without the need to recompile the system software. In response to a LISP instruction, LISP interpreter **76** calls existing subprocedures in DPS **28**, Unix Operating System **22** and/or any external application procedures which may be accessible over the network. The execution of these subprocedures implement the LISP instruction. Accordingly, the methods which can be described in LISP are limited to combinations of procedures existing in EDPS **10** or otherwise accessible to LISP interpreter **70** at the time the LISP interpreter is compiled.

In defining a new method, the user must also select the event which will trigger the execution of the new method. The following are examples of events which can be used to trigger methods (Note: certain events are available as triggers to only certain classes of objects):

1. the opening or closing of a document object,
2. the saving of a document, or automatically saving a backup version,
3. the user entering a particular content in the document associated with the procedure,
4. the user entering a particular content in the different document,
5. the starting of an external application,
6. an application outputting a particular result,
7. the mouse being moved to a certain position,
8. any test or graphic object being selected,
9. any popup menu choice of user interface **14** affecting any text or graphic object,
10. the system clock reaching a certain time or date,
11. any mouse button being pressed,
12. any key stroke or set of keystrokes, and
13. an attempt to edit any text of graphic object.

Accordingly, EDPS **10** is an extremely extensible system, allowing the user to create a variety of active documents. The following illustrates several examples of active documents which may be prepared with EDPS **10**.

1. An expense form that automatically inserts its total into a separate annual tally; the entries in the annual tally are automatically hyperlinked back to their original expense reports.
2. An interactive training manual that customizes its next chapter based upon your score on a test at the end of the previous chapter.

3. A repair manual that shows an on-screen video of the procedure in question when you press a "button".
4. A report on how a programming project is proceeding which automatically imports data from the computer-aided software engineering (CASE) application. The report has hyperlinks to the CASE software automatically installed.
5. A press release that knows to print "Draft" across its pages until the system clock reaches the announcement date listed at the beginning of the text.
6. A confidential letter of intent that won't print at all until it has been approved by all the appropriate people.
7. A proposal that shows and hides different portions depending on the security clearance of the person viewing it.
8. An urgent memo that integrates with voice annotation software so when it mails itself to someone, it announces its presence audibly.
9. A document created from a database which updates the database if you make any changes in the imported document.
10. A corporate policies bulletin that automatically formats itself to the preferences of the user who opens it.
11. An animated on-screen "slide show" presentation that tailors itself to the audience's needs based upon their response to questions.
12. A document that will let you enter text but not cut paragraphs unless you have the appropriate access.
13. A time-critical document that knows when it's due and appears on screen to remind you of when you need to work on it.
14. A calendar that knows the date, knows which documents are due, and automatically opens the ones you need to work on.
15. A document that lets you type in directions (e.g., "Go north 1.5 miles") and automatically generates a graphic map.
16. A capital request form that automatically opens up the appropriate forms as you tell it what type of equipment you need.
17. A real-estate listing, complete with text and photos, that automatically shows only the listings that meet users' needs.
18. A customer receipt that automatically runs a "frame grabber" and incorporates a photographic image of the purchaser.
19. A WYSIWYG document that can send itself over an electronic mail system.
20. A stock-market information center that retrieves stock information and displays it in text and charts that can be automatically distributed or incorporated in other documents.
21. An interactive novel, including text graphics, that completes itself based on users choices. If the user goes back to a previous selection and decides to pick up the phaser instead of the magic bow, the document re-creates itself to reflect that choice.
22. Training manuals that can be made funny, serious, sarcastic or philosophical based on reader discretion.
23. A manual that queries the reader about how much time she or he has and condenses itself accordingly.
24. Memos that send themselves to you reminding you of important dates and events.
25. A document linked to a corporate database that automatically creates a hyperlinks to an organization chart.
26. A document that creates graphic overheads formatted strictly to corporate standards. For example, if a user tries to add a sixth bulleted item, he will be told that only five are allowed on a page.
27. A speech that automatically includes reduced versions of the associated slides.
28. A weekly report that builds a time-line based upon information in a corporate database. If anyone is more than a week behind, it queries the database for more information and builds a document on a manager's desktop.
29. An international set of guidelines that knows to convert to the appropriate units of measurement—where it says "1 inch" in the U.S. version it would say "2.54 cm" in the European version.
30. A document that knows to re-send itself to the company lawyer if anyone makes any changes to it after the lawyer has approved it.
31. A scholastic test that includes an entire database of questions. The teach selects which areas to cover and how hard the questions in random order (It also tracks which questions have been used on previous tests.) It also automatically creates the answer key.
32. A phone directory that can call the person who's name you've selected; if the line is busy, it puts you into electronic mail.
33. A section of a manual that is linked to the database of standards for that industry; it won't let you make errors in formatting.
34. An on-screen brochure questions the user about her or his software/hardware configuration needs. As the user answers the questions, a suggested configuration is illustrated and listed in tabular fashion.
35. An interactive form that walks down a decision tree and shows you a graphic representation of the tree as you do so.
36. An interactive form to help you find the automobile of your dreams. It asks you for your preferences, queries a database, presents a list of the cars that meet your requirements, and then re-queries the database to build a personalized text and graphics spec sheet for the car you're interested in.
37. A contract that lists on screen everyone who has viewed it or modified it.
38. An illustrated parts catalog that builds an order form based on which items you click your mouse on—and then electronically submits your order.
39. A classified ad newsletter that builds itself automatically out of data in a database.
40. A price list that automatically converts the monetary units to the proper ones for the country in which it is being used.
41. A catalog that contains a table of options for a car with prices and specifications. As you select options, an illustration of the car updates on screen to reflect the new configuration.
42. An electronic datebook which checks the datebooks of everyone else in a workgroup and finds a time when all can meet. Any documents "pasted into" the datebook at that meeting time are automatically distributed to everyone invited to the meeting.

43. Whenever any sales region falls behind in its projections, a document is automatically built and sent to the VP of Sales. The document draws upon data in databases across the country. It automatically generates a map of the country, color-coded for how well each is performing.
44. A proposal that contains a table created from data in a spreadsheet. If any of the numbers in a particular set of columns is negative, the document queries another application, gets the explanation, and creates a footnote.
45. A document that can be open on two desktops; as each person works on it, the other person's copy is updated.
46. An automobile operating manual that has illustrations that show the steering wheel on the left or the right depending on whether the document is being viewed in the United Kingdom or in France.
47. A research report form that knows, when it's saved, to break itself into pieces and update various laboratory-wide databases.
48. A restaurant menu with actual restaurant costs there but not printed. The restaurant manager selects the items she or he wants to combine into the daily specials and reports on the restaurant's actual costs, to help the manager price the specials.

Additions, subtractions, deletions, and other modifications of the preferred particular embodiments of the invention will be apparent to those practiced in the art and are within the scope of the following claims.

What is claimed is:

1. An electronic document processing system comprising:

a document processor for creating an electronic representation of a document having one or more portions, the documents being arranged in a hierarchical structure, each portion being on one of multiple levels of the hierarchical structure;

a computer procedure creator, responsive to user inputs, for creating a computer procedure and for associating the created computer procedure with one or more user selected portions on one or more of the levels; and

a program definer for defining the created computer procedure to be operative with respect to the electronic representation of one or more portions of the document in response to at least one predetermined event.

2. The system of claim **1**, further comprising an interpreter for interpreting the created computer procedure in response to the predetermined event.

3. The system of claim **2**, wherein the interpreter includes for each class a table that identifies a location of procedures associated with the class and the objects in the class.

4. The system of claim **1**, wherein the portions of the document are objects, at least some of which are stored in a single document file, and wherein the computer procedure creator causes the created computer procedures to be stored in a procedure file separate from the document file.

5. The electronic document processing system of claim **2** wherein the interpreter is a LISP interpreter.

6. The electronic document processing system of claim **5** wherein said program definer comprises

a program editor for creating LISP procedures, and

a LISP debugger for testing the LISP procedures.

7. The electronic document processing system of claim **2** further comprising an operating system, and in which said interpreter causes the defined computer procedures to be implemented on the associated one or more portions of the document by executing subprocedures of said operating system.

8. The electronic document processing system of claim **7** wherein said operating system is a UNIX operating system.

9. The electronic document processing system of claim **2** further comprising a network interface means for interfacing with a network and for enabling said interpreter to execute the procedures by executing subprocedures stored on at least one remote device connected to the network.

10. The electronic document processing system of claim **1** wherein said at least one event includes the opening of a document object.

11. The electronic document processing system of claim **1** wherein said at least one event includes the entry of a specified object in a document object.

12. The electronic document processing system of claim **1** wherein said at least one event includes the execution of a specified application.

13. The electronic document processing system of claim **1** wherein said at least one event includes the output by a specified application of a specified result.

14. The electronic document processing system of claim **1** wherein said at least one event includes the movement of a cursor to a specified position.

15. The electronic document processing system of claim **1** wherein said at least one event includes the selection of a specified object.

16. The electronic document processing system of claim **1** wherein said at least one event includes the selection of a specified menu option.

17. The electronic document processing system of claim **1** wherein said at least one event includes the occurrence of a specified moment in time.

18. The electronic document processing system of claim **1** wherein said at least one event includes the pressing of a specified key.

19. The electronic document processing system of claim **1** wherein said at least one event includes an attempt to edit a document object.

20. The electronic document processing system of claim **18** wherein said specified key is a mouse button.

* * * * *

ELSEVIER

Knowledge-Based Systems 11 (1998) 15–23

Knowledge-Based SYSTEMS

Integrating user interface agents with conventional applications

Henry Lieberman

Media Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Received 19 May 1998; accepted 1 June 1998

Abstract

In most experiments with user interface agents to date, it has been necessary either to implement both the agent and the application from scratch, or to modify the code of an existing application to enable the necessary communication. Instead, we would like to be able to ‘attach’ an agent to an existing application, while requiring only a minimum of advance planning on the part of the application developer. Commercial applications are increasingly supporting the use of ‘application programmers’ interfaces’ and scripting languages as means of achieving external control of applications. Are these mechanisms sufficient for software agents to achieve communication with applications? This paper reports some preliminary experiments in developing agent software that works with existing, unmodified commercial applications and agents that work across multiple applications. We describe a programming by example agent, *ScriptAgent*, that uses a scripting language, Applescript, to record example procedures that are generalized by the agent. Another approach is examinability, where the application grants to the agent the right to examine internal data structures. We present another kind of learning agent, *Tatlin*, that compares successive application states to infer interface operations. Finally, we discuss broader systems issues such as parallelism, interface sharing between agent and application, and access to objects. © 1998 Published by Elsevier Science B.V. All rights reserved.

Keywords: Agents; Scripting languages; Programming by example; Programming by demonstration; Machine learning; User interface

1. Peaceful co-existence between knowledge-based systems and conventional applications

As knowledge-based systems become more sophisticated, and as people increasingly try to integrate knowledge-based systems that incorporate representation, reasoning and learning with more conventional software, the interface between the knowledge-based applications and conventional software becomes an increasingly important issue. Nowhere are these issues more critical than in software that tries to act as a *user interface agent* [14]. User interface agents offer active assistance in the user interface, providing context-sensitive suggestions, learning and predicting the needs of the user, or automating repetitive or semi-repetitive procedures. These knowledge-rich agents need to use conventional applications as if they were tools, using the application to access, edit and display objects in the domain of interest. It is not just a problem of transfer of data from the application to the agent; users wish to retain the ability to use a conventional interface to edit the data interactively themselves, as well as to delegate handling the data to an agent. However, conventional applications such as text editors, graphic editors, mail systems, CAD systems, etc. have typically been designed to be operated in real time by a live human user, not by an external program, agent or otherwise. What’s a poor agent implementor to do?

Currently, the agent implementor is faced with the following choices.

- *Don’t rely on an external application*. Implement the interactive interface that edits the data from scratch in order to be able to work with the agent. This is the most reliable option, but obviously entails the most work. If the users already have an application that they like for editing the data, they may be unwilling to give it up.
- *Modify an existing application to work with the agent*. This is next best. However, the agent implementor may not have access to the source code for the application, or it may be difficult to change it and debug changes. New versions of the application will necessitate new modified versions.
- *Use an API* (application programmer’s interface). When the application provides such an interface, this can be a good choice. However, many APIs don’t provide enough access to the underlying data structures, user interface, and other facilities of the application. As ‘agent consciousness’ grows among application implementors, this option will become better.

0950-7051/98/$ - see front matter
PII: S0950-7051(98)00056-2

- *Use a scripting language*. This is like an API, but, rather than granting the agent the right to call application routines directly, a program is written in an interpretive language that runs independently of the application. Examples are AppleScript and Visual Basic.
- *Do explicit data transfer between the agent and the application*. Communicate between agent and application explicitly by using the data transfer facilities of the application, such as writing files the agent can read or using sockets or proxy servers. This is the least desirable option.

At the Agents Group of the MIT Media Lab, over the past several years, we have explored all of these options. Mondrian [12] implemented a graphic editor from scratch in order to experiment with an agent that learns new procedures by example. Maxims [14] was a memory-based reasoning agent for e-mail filtering, implemented by modifying source code for the Eudora mail reader. Letizia [13] used the scripting language AppleScript to access the API of the commercial application Netscape, to implement a 'reconnaissance agent' for Web browsing.

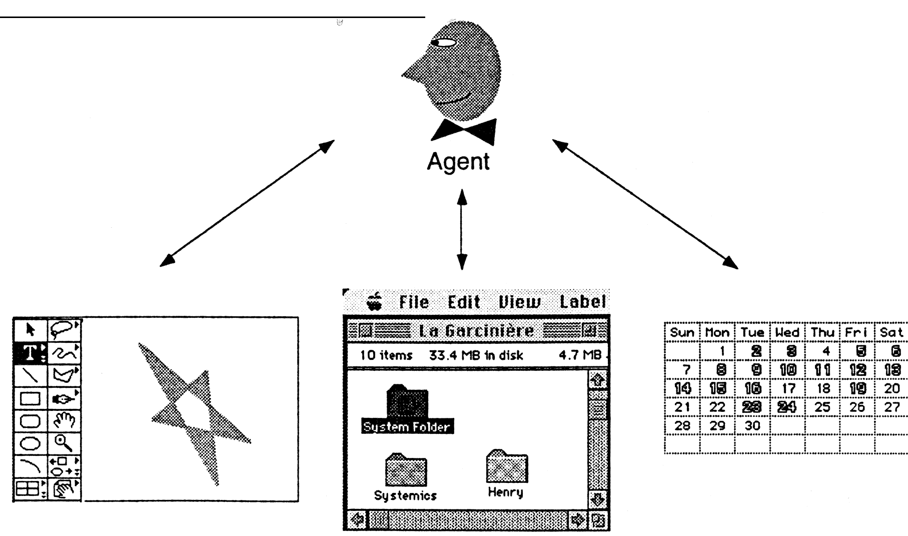
This paper reports on several experiments to use current commercial inter-application communication mechanisms to implement application-independent agents. We focus on agents that learn from observing user actions in the interface and produce generalized procedures capable of automating tasks for the user. These communication mechanisms fall short of full agent–application cooperation, but have supported some interesting experiments. We present these results, not as the definitive solution to these problems, but as lessons from experience that will help those grappling with similar problems.

The first, *ScriptAgent*, uses a so-called scripting language to record procedures and control applications. The second, *Tatlin*, uses a concept we call *examinability*, to learn from comparing successive states of the application. The point is not to provide complete descriptions of these systems, but to focus on the aspect of how these systems communicate with their target applications.

The rest of the paper surveys conceptual issues in agent–application communication. These divide into systems-oriented issues, such as access to objects and parallelism, and more high-level issues such as the sharing of the user interface between the agent and the application, how the agent stores knowledge about the application, and the division of problem-solving effort between the agent and the application in solving the user's task. Complete answers to these higher level issues remain an open research question.

2. From 'applications' to agent environments

First, we need to reconsider the idea of an 'application' itself. The dominant paradigm in commercial software for personal computers has been that of a so-called 'application', a self-contained, shrink-wrapped product bought from a single supplier to do a well-defined task. The computer user chooses among the available applications on the machine, 'enters' an application, does some focused work on a set of documents manipulated by that application, then 'leaves' the application and may enter a new application. Each application has its own interface, and moving between applications means interacting with a different (but possibly overlapping) interface. Recent operating systems have improved upon this only slightly by allowing several applications to be 'open' at once, or to let a document produced by one application be 'contained' in a document produced by another application, as in OpenDoc or Microsoft OLE.



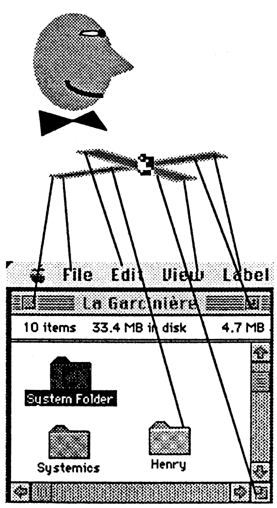
Applications

The paradigm of thinking of software in terms of self-5contained and isolated applications or documents is becoming rapidly obsolete. Computing environments are

getting more complex, and users are getting tired of the artificial barrier between applications. Users want to work with text, graphics, communications, programming, etc. seamlessly in an integrated environment. However, since this is difficult to achieve in current environments, agents that act as intermediaries can provide a stopgap. Agents can be seen as a way of supplying software that acts as the representative of the user's goals in the application environment. Agent software can provide glue between the applications, freeing the user from the complexity of dealing with the separate application environments.

3. Attaching agents through the user interface: 'marionette strings'

Efforts are now underway in the commercial arena to



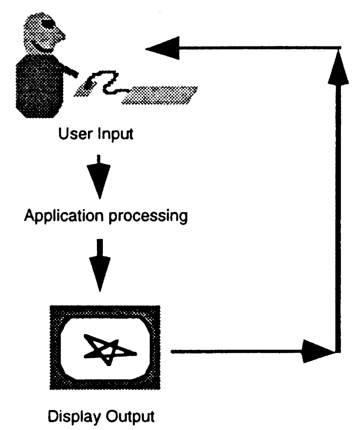
''Marionette strings''

put together some sort of inter-application communication. The intent of these efforts is primarily to support inter-application data-sharing, scripting (manually writing external programs to control an application). The best of them provide simple macro-recording. However, developers of such languages also envision that they might serve as support for future interface agent software as well. Examples of such efforts include AppleEvents and AppleScript, Microsoft OLE, Com/DCom, ActiveX, and Visual Basic, TCL, Java, Javascript, and others.

Many applications such as Excel, Emacs, Director, Netscape and others have an embedded programming language in the application. Such languages have primitives that allow direct access to the objects and functions of the application but, unfortunately, do not provide a direct way of interacting with other such applications.

Scripting languages such as AppleScript and Visual Basic, which are intended for use across applications, take advantage of the following observation. Regardless of programming language or implementation, most modern direct manipulation interfaces are structured around an interpreter called an 'event loop' that accepts mouse and keyboard input, tries to determine what functionality the input is requesting, changes the application data structures, and updates the screen display. Many UIMS systems supply at least a rudimentary form of this interpreter.

Although most commercial applications are not written so that their functionality is accessible by another program,

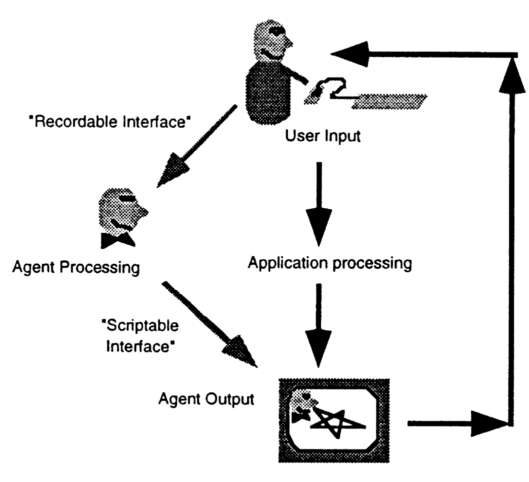


The event loop in an interactive application

they must provide some way for the user to access that functionality through input, typically by selecting some item from a menu, clicking on some icon, or typing some command. Thus, if another program can 'fool' the application into thinking that the user typed or clicked something, it too could access the functionality of the application.

I call this approach 'marionette strings' because the agent is given a set of 'strings' corresponding one-to-one with user actions in the interface, and can 'tug' on the strings to make the program perform. The marionette strings are implemented by stuffing an object representing user input into the application's input buffer. The application then processes it as if the input was produced by user interaction.

This works, after a fashion. Because there is some degree



Event loop with agent participation

of standardization in interfaces for invocation of user-level commands, such as pull-down menus, or double-clicking on icons, it is easier to get different applications to agree on a standard format for input than for how to invoke the underlying functions. (If there were no interface standardization, the users would rebel!)

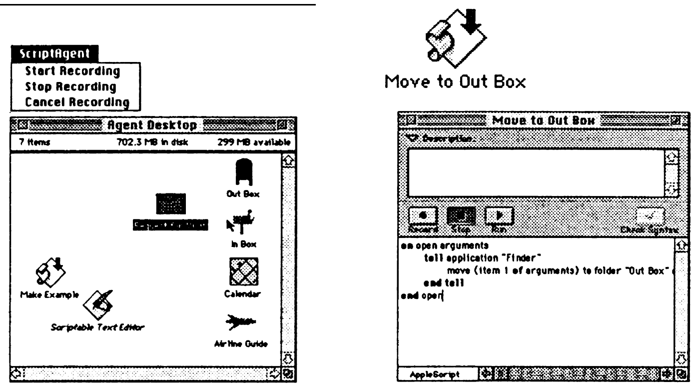
But it is less effective for returning values from operations, or for accessing application data. From the user's point of view, the result of the operation is simply visible in the graphical interface. An agent program cannot 'see' what the user sees (though see [18] for an approach that actually parses the screen display!). Other operations that are necessary for a program but do not have direct correlates in the interface are also difficult.

One approach is to provide more marionette strings by permitting 'virtual events' [9] that have no direct counterpart in the interface but are treated as events in the input buffer. This introduces some interpretation overhead, and may complicate the operation of the interpreter.

4. Scriptagent: a learning agent based on a scripting language

As an example of how we might use the marionette string approach to implement an agent, we implemented ScriptAgent, a programming by example system based on Apple's AppleScript inter-application scripting language. ScriptAgent uses techniques from the author's Mondrian system [12], which learns procedures in a graphical editor. ScriptAgent's initial domain is Apple's Scriptable Finder, but it is targeted for use with any AppleScript scriptable and recordable application.

ScriptAgent is one of the few attempts to build a learning agent that works with standard operating system software and unmodified commercial applications. This should greatly extend the scope and practicality of agent learning systems. ScriptAgent's use of AppleScript also enables it to be one of the few agent systems that could operate across multiple applications. Macro recorders such as Apple Macromaker or CE Software's Quickeys do provide recording and replay of unmodified input events across applications, but no generalization of the recorded procedures. ScriptAgent also serves as one of the first serious tests of AppleScript's original intention to provide monitoring of user interface actions and effective control of applications by an external agent.



ScriptAgent: The Finder interface (left) and generated application and script (right)

5. Integrating the agent's interface with the application's interface

For the Finder, we have tried to integrate the agent's interface into the Finder's interface itself. We use AppleScript-generated applications to represent the interface with the agent. Procedures recorded by the agent are stored as AppleScript applications that are invoked by double-clicking or dragging as other applications are. ScriptAgent's generalization operation is represented by an application Make Example.

Dragging a Finder object onto the Make Example icon says that the object is to be thought of as an example for the procedure that is being taught to the system. All subsequent user operations are recorded as relative to that example. If the example object is used as an argument to further operations, then those operations will depend upon the example as well, and the next time the procedure is executed the new objects will be substituted.

5.1. An example

As a very simple example, we show a simple file manipulation procedure in the Finder. We pick a specific file named ''Request for Visit'' to serve as an example. We will demonstrate the procedure on Request for Visit and the system will generate a procedure that can be applied to any file. We turn on recording, drop the file onto the Make Example application, then demonstrate the procedure. On the right of the illustration is the AppleScript code as generalized by ScriptAgent. This generates an AppleScript application which, when a new file is dropped on it, performs the analogous Move and Duplicate actions.

The Make Example operation is duly recorded in the AppleScript code recorded by the Scriptable Finder, and is

therefore integrated into the procedure being recorded. It is noticed by ScriptAgent's generalization code, which designates the argument to Make Example as a generalization.

5.2. Scriptability

The term scriptable (following Apple terminology) is used if the application provides a means (either through a scripting language or through a so-called application program interface [API]) for an external agent to invoke the commands of the application. To be fully scriptable, the application must allow the external program to invoke any function that the user can initiate by selecting from menus, clicking on icons, or typing. An application will be called recordable if it is capable of reporting to an external agent when the user asks the application to perform a function, by menu or icon selections, or by typing.

Making an application fully scriptable and recordable is a very strict requirement. Several years after Apple introduced AppleScript scripting and asked application developers to comply, sadly, few developers have done so. Many applications are at least partially scriptable, but few provide either a complete scripting interface or a usable recording capability. The situation on other platforms is similar. A commercial interface agent using neural network learning techniques [2] has had to resort to patching system routines in order to observe and affect user interface actions, and was ultimately unsuccessful. However, we do hope that the number of scriptable and recordable applications will increase, and the existence of intelligent interface agents will certainly increase the incentive for developers to provide external control. Though 'scriptable' is in its name, it is actually recordability which is of more interest for programming by example, and is the reason why ScriptAgent operates in the Finder's domain.

parser and unparser from AppleScript to CLOS objects, using Joachim Laubsch's Zebu [11], an LALR parser compiler.

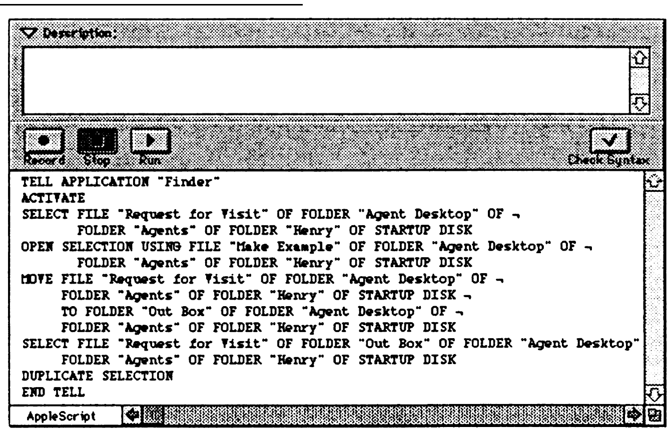
The approach of reusable parsing/unparsing technology is important. We are entering an era where the computer environment will have many co-existing scripting languages or other text-based application-specific languages. AppleScript, Java, JavaScript, TeleScript, Visual Basic and TCL are all examples of this trend. No one of these languages is computationally general enough to single-handedly support the kinds of agent applications we are envisioning, and agent programs may have to deal with code written in several of them. Parsing the various representations into a single dynamic environment suited for symbolic computation seems like a strategy that will cope with a multilanguage world.

ScriptAgent, like many agent programs, is a program-writing program, so it must be able to analyze, generate and execute data representing program code. Generation of code for agent actions is accomplished by methods that act as code walkers or partial meta-interpreters on the parsed representations. For ScriptAgent, these methods perform the explanation-based generalization of recorded procedures.

5.4. Describing actions and objects

The basic mechanisms for recording and generalizing procedures in ScriptAgent are taken from Mondrian [12] and consist of an explanation-based learning mechanism that generalizes on user-designated example objects and propagates generalizations through structures that record the dependencies between operations.

The illustration below shows the user interface actions as originally recorded by AppleScript's default action recorder. Note that specific file paths appear where ScriptAgent has generalized an argument to the resulting application.



Applescript's recording of user interface actions.

5.3. Enabling the agent to work with AppleScript

Understanding and generalizing the AppleScript code as it comes from the Scriptable Finder is done via a

How the user interface recording mechanism describes objects involved in user interface operations is known in the field of programming by example as the *data description problem* [3]. This problem is central to how the agent and

the application will cooperate. Different choices in how to describe objects will result in the agent having different ideas about 'what the user did'.

In AppleScript, objects are described using AppleScript expressions which designate a 'path' to the object of interest, e.g. ''Window 2 of Document 1''. The choice of describing the object in this way (as opposed to, say, the window named ''Foo'', or the last window the user selected) is made by the recording mechanism rather than by the agent. Nothing in the agent can affect the way this choice is made.

Unfortunately, AppleScript does not give the agent direct access to the objects which are the arguments to and the results of operations themselves. Were it to do so, the agent could construct its own data description of the object, as occurs in most programming by example systems [3]. However, AppleScript cannot do this, because it does not require the application to provide a full object model for all its internal data. The applications may be hard-coded C programs which may not be able to deliver pointers to particular internal data structures. If the user selects a word in a word processor, the word processor application may or may not be able to return ''Word 3 of Line 20 of Document 2'' as a value that can be relied on in subsequent operations. So-called AppleScript Objects are not required to have a lifetime beyond a single AppleEvent transaction [7].

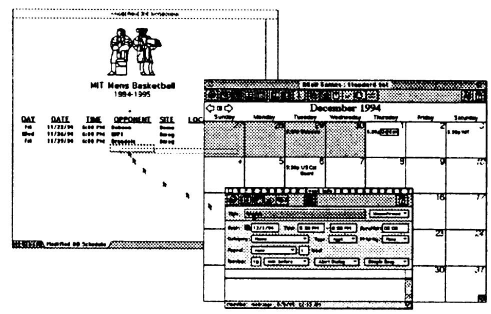
For the Scriptable Finder, ScriptAgent deals with this problem by using the file system to dereference expressions, since all the objects in the Scriptable Finder's domain (files, applications, etc.) are accessible directly to ScriptAgent through the file system. However, for an arbitrary application, it may be necessary for the agent to keep 'shadow' objects to model the application's data. In the worst case, the agent may need to mimic the application's functionality in order to access the result of an operation.

5.5. Examinability

Because scriptability is so much more prevalent than recordability, we are also exploring an alternative approach which we call examinability to allow an agent to operate when full recordability is not available. We notice that, as part of scriptability, many applications do give an external agent the right to examine internal data structures. Thus we can have an agent poll the application data structures and try to infer the user interface actions by comparison of successive application states.

This approach is taken in Tatlin, a system by my student David Gaxiola [6], which integrates a commercial spreadsheet (Microsoft Excel) with a calendar program (Now Up-to-Date). The user interactively transfers an example entry from the calendar to the spreadsheet, and the agent learns a procedure that can perform an analogous transformation on similar entries, using similarity-based learning.

In this scenario, a user has a series of sports events noted in a calendar, and wishes to construct a spreadsheet summarizing information about the events. The user demonstrates a single example of how to transfer the information from an appointment in the calendar to appropriate columns of the spreadsheet, using Cut and Paste operations. The agent is then asked to generalize a program that can be given a set of events in the calendar and have them automatically transferred to the spreadsheet.



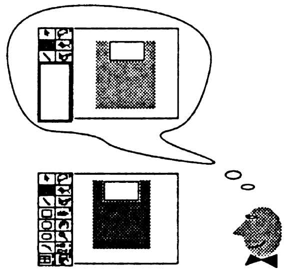
Tatlin can infer transfers of data from examples

The agent examines the internal state of the calendar application, starting from the appointment window currently open and selected. Tatlin has an object model for the calendar application, and reads the various fields of the appointment data structure. The user must then select the destination data structure in the spreadsheet (in this case, a row) and Tatlin tries to match up the fields. Tatlin can accept advice to pay attention (or not) to certain attributes of data, such as fonts, case, or numeric format.

Tatlin assumes that fields that have similar names or contents are causally connected by user interface actions, such as cutting the Date field from the event and placing it in the Date column of the spreadsheet entry. Such an assumption can be erroneous, of course, but the user is assumed to be providing 'good' examples that do not have collisions, in order to clearly teach the agent. A practical problem that arises in such systems is making these inferences robust in the face of minor details such as differences in time and date formats, and differences in terminology between applications (one may refer to a document's 'Title' while another may refer to its 'Name').

5.6. An agent's 'mental model' of the application

Both ScriptAgent and Tatlin bring up the issue that it is necessary for the agent to have an object-oriented model of the data in the application it is dealing with. Since the agent is supposed to act as a proxy for the user, one can think of this model as the agent's 'mental model' of the application. As in a user's mental model of the application, the model need not consist of all the objects in the application, but at least those that are directly visible and of concern to the user



An agent's ''mental model'' of an application

for specific purposes in the interface, and the minimal set of internal objects that affect the application.

A problem for programming an agent is that the objects internal to the application are not accessible directly to the agent. The agent must thus deal with foreign objects. Foreign objects may be written in a language different from the language of the agent, they may be in another process, or they may be stored on another machine on a network. Several proposals are currently under consideration for how Java might interact with foreign objects stored in other locations on the Web.

We have implemented a foreign object interface in CLOS (Common Lisp Object System). This is a facility that allows a calling program to access foreign objects more or less transparently, as if the objects were local to the calling program. It works by keeping a table of objects in each local program that are referenced by foreigners, managing that table on an as-needed basis. The foreign object interface is intended to be complementary to so-called foreign function interfaces, such as remote procedure calls or inter-language function calls. Some foreign function interfaces do provide translation of basic data types such as numbers and strings, but complex objects that may point to other objects are not typically provided for. Leaving object management up to the applications programmer is typically too great a burden, and effectively discourages routine use of multilanguage or multiprocess programs. The return value of each foreign procedure call is a foreign object, a representative of an object in another application. Functions applied to the foreign object, and requests to access the components of the foreign object, result in forwarding the request to the server application.

It is important that foreign objects are created in a lazy or on-demand manner. The client program need not have a foreign object corresponding to every object in the server. Foreign objects for compound objects on the server are created on the client only to the depth to which they are referenced, and not in their entirety. This property is essential for making the foreign object interface efficient enough to be practical. It is also important that access to the object be transparent. If it is not, the caller must be prepared to deal with the possibility that any object might, in fact, be foreign, which complicates calling interfaces.

A request to create a new object on the server is handled by a process analogous to what is called interning in interpretive systems. Interning is a process that translates strings into references to objects. It involves keeping a table of objects, usually in the form of a hash table. The first time a string is encountered, a new object is created and stored in the table. The next time a string containing the same sequence of characters is encountered, the original object corresponding to that string will be returned.

Each server maintains a registry, a table of objects that are referenced from outside. When a server receives a request to create a new object, it makes a new entry into the registry and returns an index into the registry. The foreign object on the client contains the server connection and the index into the registry. The metaobject protocol [8] can be used to achieve transparency between the use of foreign and local objects. The system exchanges objects with a Lisp running on another machine using the Apple Events [7] inter-application communication protocol. We have not dealt with the problems of garbage collection across multiple address spaces; once an object is entered into the registry it remains forever. There are several proposals for algorithms for network garbage collection in the literature. Inter-language garbage collection, also desirable in the general case, is also not handled in our implementation.

6. Issues in agent–application integration

6.1. Granularity of the event protocol

For the marionette string approach to work, the application writer must agree on a set of operations to 'export' as events, which determines what is accessible to the agent, and the agent must agree to 'import' the events. This is especially important in approaches like Kosbie's [9], where virtual events are created that do not correspond directly to explicitly visible user interface actions.

There is the problem of deciding at how fine a level to do this. In the limiting case, every time the program does anything, it must create an event corresponding to that action, and be prepared to execute that action in response to an event requesting it from the agent. In that case, the application effectively becomes a program written in the event language rather than the underlying programming language. The 'application code' itself becomes nothing but an interpreter for the code written in the event language. At worst, the interpretation overhead would slow the application, and at best this would require two versions of everything, an interpreted and a compiled version.

To take a concrete example, should we include mouse movements in the protocol that an agent should be able to receive? If we say yes, then we are obliging every application to report every time the mouse moves, which is potentially inefficient, and wasteful in the case where the agent does not need this information. If we say no, we are preventing all agents from ever tracking the mouse, and it seems that at least some agents might in fact want this ability.

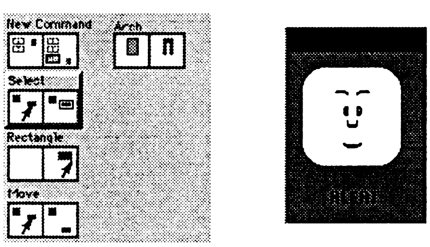
The moral is that there should be at least some provision for dynamic negotiation of a protocol between agent and application. It may not be possible to fix in advance an application protocol that will be satisfactory to every possible agent, nor an agent protocol that will be acceptable to every application.

6.2. Sharing an interface between agent and application

Another issue is that the agent itself may need to interface directly with the user, and most applications are not currently prepared to share their interface with another program whose interface they do not know in advance.

Mondrian [12], for example, adds icons to the draw program's palette, using the same 'domino' icon style in which the draw program's operations are represented. In the illustration below, the New Command icon at the upper left is an icon that operates the agent, the three icons below it are draw operations provided by the draw program, and the Arch icon at upper right was added by the agent to the interface as a result of user interaction.

Kozierok's Calendar Agent [14] displays a cartoon face



Application icons (left column) and Agent-generated Arch icon. Right: Anthropomorphized calendar agent

whose expression indicates the state of the agent and feedback about its predictions. Cypher's Eager [3] has an anthropomorphic cat to represent the agent, and colors menu operations for anticipatory feedback.

Applications should be able to accept requests to extend their interface by adding or modifying user-accessible commands and objects, reserving parts of the screen or modifying the program's user interactions, as part of the protocol. Such requests are not typically a part of inter-application communication protocols.

6.3. Parallelism between agent and application

Parallelism is also an issue. Any user interface program is in fact a parallel program, even if there is no parallelism going on in the functions implemented by the application. This is because there are always at least two processes that are running simultaneously: the computer and the user!

This is a well-known problem in user interface programming, but it becomes worse when the actions of an autonomous interface agent are added to the system. The application and agent program may be acting concurrently on the same objects, and synchronization must be provided for where necessary. Another merit of the marionette string approach is that it achieves cheap synchronization for programs that are completely 'event driven': only one event may be processed at a time. For more complex kinds of programs, such as those which take background action while the user continues to use the interface, this kind of synchronization will be inadequate.

6.4. Related work

There are an increasing number of projects in agent software that are trying to integrate with more traditional applications, though there is still no definitive methodology for this. [16] is one of the few references that systematically discusses these issues. [19] provides a good discussion of user/agent collaboration issues, including initiative, dialogue and turn taking. [5], [9] and [17] present proposals for application-independent agent kits, based on histories of user-interface events. Some interface agents, such as [10] and [4], are relatively loosely coupled to applications with command-oriented or conversational interfaces, and don't require the tight agent–application coupling discussed here. All of these references depend, at least to some extent, upon some explicit cooperation from the target applications.

A related approach to Tatlin underlies Robo-Format [1]. Robo-Format used examinability of Excel to automatically compile formatting templates for spreadsheet cells. Earlier, Myers and Werth's Tourmaline [15] used the scripting language WordBasic to generalize formatting templates for text.

7. Conclusion: messages to three different audiences

To conclude, the purpose of this paper is to convey messages to three different kinds of audience about the potential for agent–application communication.

To *application developers*, it is a plea for providing rich and robust interfaces that can be used by agents to take advantage of your application. These might take the form of APIs and/or scripting languages, depending on your platform. Keep in mind that your program may be used not only by a human user but by an agent program, and provide the necessary interfaces for it to take advantage of your application. For each kind of object, provide facilities to create and delete the object, and access its properties. Provide facilities for each operation on the object, and provide a robust way to refer to the object from outside. Provide a means for the agent to share your user interface, and guard against the possibility of conflict by parallel changes by the user and agent.

To *platform developers*, and this includes developers of

languages, operating systems, interface toolkits and even hardware: realize that agents and applications need to co-exist on your platform. Make data structures and procedures as dynamic as possible. Lightweight and reliable interprocess communication is essential. And don't forget to give the agent and application developers some reasonable means of debugging the agent–application interaction when things go wrong!

Finally, to *agent developers*: we need to continue to do experiments in developing interesting agents and trying to hook them up to existing applications. When current facilities don't work, we need to be more articulate about why, and insist on the facilities we do need. It is a chicken-and-egg problem. We won't get agent infrastructure until we demonstrate the value of agent applications, and we can't fully get our agents working until the new agent infrastructure is there. We need to convince application and system developers that the work to provide agent infrastructure will be worth any perceived extra cost in providing it. The best way to do this is to show how agent applications will provide new value for the user, and enlist the help of the users in asking for it.

Acknowledgements

Thanks to Pattie Maes for her insights concerning intelligent agents. Max Metral has provided valuable insights into application–agent communication. Thanks to Jim Miller, Tom Bonura, Allen Cypher, Ike Nassi and Steve Strassman. All names of commercial products mentioned herein are trademarked by their respective manufacturers. Support for this work comes in part from research grants from Apple Computer, the National Science Foundation, British Telecom, Exol, IBM, the News in the Future Consortium, the Digital Life Consortium, and other sponsors of the MIT Media Laboratory.

References

[1] J. Ash, J. Schlimmer. Robo-Format: A sample self-customizing application. Washington State University, 1995. ftp: //ftp.eecs.wsu.edu/papers/schlimmer/ash-robo.ps.

[2] A. Caglayan, M. Snorrason, J. Jacoby, J. Mazzu, R. Jones. Lessons from Open Sesame! A user interface learning agent. In: Conference on Practical Applications of Agents and Multi-Agent Systems (PAAM–96), London, 1996.

[3] In: A. Cypher. Watch what I Do: Programming by Demonstration. MIT Press, Cambridge, MA, 1993.

[4] O. Etzioni. A Softbot-based interface to the Internet. Communications of the ACM, July 1994.

[5] M. Frank. Standardizing the representation of user tasks. In: AAAI Spring Symposium on Acquisition, Learning and Demonstration: Automating Tasks for Users, Stanford, CA, March 1996.

[6] D. Gaxiola. Tatlin: integrating commercial applications into programming by demonstration. MIT BS thesis, 1995.

[7] D. Goodman. Danny Goodman's AppleScript Handbook. Random House, New York, 1994.

[8] G. Kiczales, D. Bobrow. The Art of the Meta-Object Protocol. MIT Press, Cambridge, MA, 1992.

[9] D. Kosbie, B. Myers. A system-wide macro facility based on aggregate events. In: A Cypher. Watch What I Do: Programming by Demonstration. MIT Press, Cambridge, MA, 1993.

[10] A. Lansky, M. Friedman, L. Getoor, S. Schmidler, N. Short, Jr. The Collage/Khoros link. In: Workshop on AI and the Environment, IJCAI–95, Montréal, Canada, August 1995.

[11] J. Laubsch. Zebu: A tool for specifying reversible LALR(1) parsers. Hewlett–Packard Laboratories, 1992. ftp: //ftp.digitool.com/pub/MCL2/contrib/.

[12] H. Lieberman. Mondrian: A teachable graphical editor. In: A Cypher. Watch What I Do: Programming by Demonstration. MIT Press, Cambridge, MA, 1993.

[13] H. Lieberman. Autonomous interface agents. In: A Cypher. Watch What I Do: Programming by Demonstration. MIT Press, Cambridge, MA, 1993.

[14] P. Maes, R. Kozierok. Learning interface agents. In: AAAI Conference, 1993.

[15] B. Myers, A. Worth. Tourmaline: text formatting by demonstration. In: A Cypher. Watch What I Do: Programming by Demonstration. MIT Press, Cambridge, MA, 1993.

[16] A. Newell, D. Steier, Intelligent control of external software systems, AI in Engineering 8 (1993) 3–21.

[17] P. Piernot, M. Yvon. The Aide project: an application-independent demonstrational environment. In: A Cypher. Watch What I Do: Programming by Demonstration. MIT Press, Cambridge, MA, 1993.

[18] R. Potter. Triggers: Guiding automation with pixels to achieve data access. In: A Cypher. Watch What I Do: Programming by Demonstration. MIT Press, Cambridge, MA, 1993.

[19] C. Rich, C.L. Sidner. COLLAGEN: when agents collaborate with people. In: Autonomous Agents Conference, Marina del Rey, CA, February 1997.

'789 patent, claim 17	Obvious over IBM in view of Interleaf
	portion [1.4].
[17.2] at least one communications interface,	[17.2] IBM teaches that the AS/400 and the printers both have "at least one communications interface": "Print with complete system management and full error recovery, whether the printer is system-attached via twinax or network-attached via TCP/IP." (IBM, p. 1) Having the printer either "system-attached" or "network-attached" to the system shows that both the system and the printer have a communications interface so that they can be attached to each other.
[17.3] characterized in that the data processing unit is programmed in such a way that it operates in accordance with the method as claimed claim 1.	See claim portions [1.0]-[1.7]

'789 patent, claim 20	Obvious over IBM in view of Interleaf
[20.0] A printer,	[20.0] IBM teaches "a printer": "AFP helps you combine the capabilities of high-function *Intelligent Print Data Stream (IPDS) printers* and print software…" (IBM, p. 1) The IPDS printers are the printers as recited in the claim.
[20.1] characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.	[20.1] See claim portions [17.0]-[17.3].

'789 patent, claim 21	Obvious over IBM in view of Interleaf
[21.0] A printer server,	[21.0] IBM teaches "a printer server": "LAN print servers can receive print jobs from AS/400 and manage those files to either IPDS or ASCII printers." (IBM, p.

* In the context of the present request, the standard provided in MPEP § 2111 for claim interpretation during patent examination may be applied whereas a different standard may be used by a court in litigation. The PTO is not required to interpret claims in the same manner as a court would interpret claims in an infringement suit. Real parties in interest reserve the right to argue for a narrower or different construction of any term or claim in any pending or future litigation concerning this patent or any related patents.

'789 patent, claim 19	Obvious over IBM, Interleaf, Interleaf Patent, and Lieberman
	can be read or changed. See the analysis for [15.1]. The Interleaf Patent teaches that the data processing unit allows the script objects to be read or changed: "Toward this end, environment 72 includes a program editor 74 for allowing a user to define the new method in the interpretive language LISP." (Interleaf Patent, 5:53-55) Allowing the user "to define a new method" in the Lisp scripting language shows that preferably also the script objects can be read or changed. ——— [19.2c] The Interleaf Patent teaches that the application interface allows the graphically representable objects "to be read out, to be changed, to be deleted or to have new objects (5) appended": "The system allows the user to *edit the document by modifying existing text and graphics or by adding entirely new text or images*. As the user modifies the document, the display is continually updated to reflect these changes, thereby allowing the user to interact with the system to achieve the desired document." (Interleaf Patent, 1;15-27) "Modifying exiting text and graphics or by adding entirely new text or images" shows that the graphically representable objects in the application interface are able to be read out, to be changed, to be deleted or to have new objects appended as recited in the claim. See claim portion [1.4b]. As discussed relative to claim portion [19.2b], this includes the script objects.

'789 patent, claim 20	Obvious over IBM in view of Interleaf and the Interleaf Patent
[20.0] A printer,	[20.0] IBM teaches "a printer": "AFP helps you combine the capabilities of high-function *Intelligent Print Data Stream (IPDS) printers* and print software…" (IBM, p. 1) The IPDS printers are the printers as recited in the claim.
[20.1] characterized in that	[20.1] See claim portions [17.0]-[17.3].

* In the context of the present request, the standard provided in MPEP § 2111 for claim interpretation during patent examination may be applied whereas a different standard may be used by a court in litigation. The PTO is not required to interpret claims in the same manner as a court would interpret claims in an infringement suit. Real parties in interest reserve the right to argue for a narrower or different construction of any term or claim in any pending or future litigation concerning this patent or any related patents.

'789 patent, claim 1	Anticipation by Interleaf
	are input to the program, thus showing reading in the input print data stream as recited in the claim.
[1.2] (ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and	[1.2] Interleaf teaches that the input print data stream is "analyzed by means of a parser": Interleaf describes the initialization of the document file stream. "If a document contains active objects within the document file stream (as opposed to the methods file), these become activated when the document is opened programmatically or by the user for viewing, editing, or printing." (Interleaf, p. 84) The input data stream is analyzed by means of a parser when it is "activated." The objects in the stream are analyzed by a Lisp interpreter: "Lisp was chosen as the I6 extension language because it is interpreted (providing a fast prototyping environment) and has run-time binding. Run-time binding is required for easy delivery of active documents." (Interleaf, p. 77) Interleaf continues: "Lisp is a good language for active document development due to features such as the interpreter (which allows nice development tools), functions as data (it is easy to store and manipulate programs as data on document objects), and run-time binding (which is important for the delivery of active document objects)." (Interleaf, p. 85) The "interpreter" is the parser as recited in the claim. The "run-time binding" of the "active document objects" by the Lisp interpreter shows that the data stream is analyzed by means of a parser as recited in the claim. Interleaf shows the parsing and analysis of the objects during activation: "The system contains two facilities to limit activation.... The second is to define an "active load hook" that gets called prior to each automatic activation. *It is passed the current object and the code that is about to be evaluated. A sophisticated program can examine this code and decide whether or not to permit its activation. We note that examination of code by code is easily*

* In the context of the present request, the standard provided in MPEP § 2111 for claim interpretation during patent examination may be applied whereas a different standard may be used by a court in litigation. The PTO is not required to interpret claims in the same manner as a court would interpret claims in an infringement suit. Real parties in interest reserve the right to argue for a narrower or different construction of any term or claim in any pending or future litigation concerning this patent or any related patents.

'789 patent, claim 1	Anticipation by Interleaf
	done in Lisp, which makes no distinction between code and data. A simple example of such an inspector is distributed with the system." (Interleaf, p. 85) When a document file stream is loaded, the "active load hook" examines the code objects that are about to be activated. This shows the Lisp interpreter (the parser) analyzing the input document file stream as recited in the claim. Interleaf teaches parsing the input print data stream "for graphically representable objects." Interleaf teaches that the objects in the input print data stream are graphically representable objects: "In using *document objects to represent UI elements*, there are two common approaches used *to change their appearances* in I6 documents. First, one can programmatically apply different graphic properties or transformations to the object, such as changing its color or size or position. Second, one can use the conditional document assembly mechanism [18] to hide one document object quickly (such as an empty box) and instead show another object (such as a filled box)." (Interleaf, p. 81-82, emphasis added.) The "document objects" are the graphically representable objects as recited in the claim. Having the "document objects" "change their appearance" (including "color or size or position") shows that the document objects are graphically representable as recited in the claim. These document objects make up the document: "The most important element in an active document system is the underlying structured document model. It should contain a rich set of document objects (*text, components, tables, autonumbers, page numbers, index tokens, reference objects, footnotes, frames, diagrams, beziers, images, files, documents, books, etc.*), and a rich set of properties for each object." (Interleaf, p. 83, emphasis added.) The "text, components, tables, autonumbers, page numbers, footnotes, frames, diagrams, beziers, images… etc" are all examples of graphically representable objects as recited in the claim.

* In the context of the present request, the standard provided in MPEP § 2111 for claim interpretation during patent examination may be applied whereas a different standard may be used by a court in litigation. The PTO is not required to interpret claims in the same manner as a court would interpret claims in an infringement suit. Real parties in interest reserve the right to argue for a narrower or different construction of any term or claim in any pending or future litigation concerning this patent or any related patents.

'789 patent, claim 19	Obvious over the Interleaf Patent in view of IBM
	user to interact with the system to achieve the desired document." (Interleaf Patent, 1;15-27) "Modifying exiting text and graphics or by adding entirely new text or images" shows that the graphically representable objects in the application interface are able to be read out, to be changed, to be deleted or to have new objects appended as recited in the claim. See claim portion [1.4b]. As discussed relative to claim portion [19.2b], this includes the script objects.

'789 patent, claim 20	Obvious over the Interleaf Patent in view of IBM
[20.0] A printer,	[20.0] The <u>Interleaf Patent</u> is directed to "printing". (Interleaf Patent, 1:9) <u>IBM</u> teaches "a printer": "AFP helps you combine the capabilities of high-function *Intelligent Print Data Stream (IPDS) printers* and print software..." (IBM, p. 1) The IPDS printers are the printers as recited in the claim.
[20.1] characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.	[20.1] See claim portions [17.0]-[17.3].

'789 patent, claim 21	Obvious over the Interleaf Patent in view of IBM
[21.0] A printer server,	[21.0] It would be obvious to implement the system of the Interleaf Patent as a "print server." Running printer software on a printer server is "merely a matter of obvious engineering choice," and thus *prima facie* obvious. (MPEP § 2141.04.V.(B)). The '789 patent suggests no unexpected or unpredicted benefit of running the claimed method on a print server, and in fact merely refers to a print server as an alternative implementation without any further description. ('789 patent, 8:24-25). <u>IBM</u> also teaches "a printer server":

* In the context of the present request, the standard provided in MPEP § 2111 for claim interpretation during patent examination may be applied whereas a different standard may be used by a court in litigation. The PTO is not required to interpret claims in the same manner as a court would interpret claims in an infringement suit. Real parties in interest reserve the right to argue for a narrower or different construction of any term or claim in any pending or future litigation concerning this patent or any related patents.

A first action notice of allowance was mailed (10/07/2003) for Application Control 10/275,784 and issued as USPN 6,684,789 (02/03/2004). Reasons for allowance recite: "Claims 1-53 have been indicated for allowance because the prior art fails to teach the entire combination of a method for the transformation of digital print data stream including a steps of reading an input print data stream, analyzing the input data stream by means of a parser and splitting the input data into the graphically representable objects, storing the graphically representable objects in a memory in an object-oriented format, transforming the object-oriented format into a format for controlling a printer, the object-oriented format stored in the memory including at least one stored script is assigned, which is executed in the case defined in the script."

Substantial New Question (SNQ) Of Patentability Proposed by Requester

The Request indicates that the third party requester considers the following substantial new question of patentability (SNQs) to be raised by the prior art cited in the Request:

SNQ. The third party requester considers a substantial new question of patentability as to claims 1-53 to be raised by the "IBM" reference alone or in combination with one or more secondary prior art references: "Interleaf", "Interleaf Patent" and/or "Lieberman.

The substantial new question of patentability with respect to "IBM", "Interleaf", "Interleaf Patent" and/or "Lieberman" is based on new teachings, not previously considered or addressed in the prior examination of the patent or of a final holding of invalidity by the Courts. The above noted prior art was not cited or used in a rejection during prosecution history. The "IBM" reference teaches all of the claim elements that were listed in the Reasons for Allowance as

missing from the prior art of record during the original prosecution of the '789 patent. "IBM" is therefore new and non-cumulative of the prior art originally of record: "IBM" teaches reading an input print data stream (p. 193), analyzing the input data stream by means of a parser (p. 194), splitting the input data into the graphically representable objects (p. 194), storing the graphically representable objects in a memory in an object-oriented format (p. 7), and object-oriented format stored in the memory including at least one stored script (p. 127; 134-135). To the extent that "IBM" fails to explicitly teach LISP scripts, "Interleaf" teaches reading an input print data stream (p. 84), splitting the input data into the graphically representable objects (p. 85), storing the graphically representable objects in a memory in an object- oriented format (p. 82), including at least one stored script (Lisp scripts) (p. 77). To the extent that "IBM" fails to explicitly teach script commands, the "Interleaf Patent" teaches many of the same scripts found in the '798 dependent claims (columns 6-9). Lieberman is relied upon for expressly teaching the use of Javascript as a scripting language (p. 17). There is a substantial likelihood that a reasonable examiner would consider these teachings important in deciding whether or not the claims are patentable. Accordingly, the "IBM" reference, taken alone or in combination with "Interleaf", "Interleaf Patent" and/or "Lieberman", is considered to raise a substantial new question of patentability as to claims 1-53 of the '789 patent.

Conclusion

For the reasons above, claims 1-53 of USPN 6,684,789 B2 to Krautter will be reexamined.

All correspondence relating to this *inter partes* reexamination proceeding should be directed:

By EFS: Registered users may submit via the electronic filing system EFS-Web, at https://sportal.uspto.gov/authenticate/authenticateuserlocalepf.html.
By Mail to: Mail Stop Inter Partes Reexam

object oriented system including electronic document elements such as overlays, fonts, and images.

Regarding **claims 17 and 20**, IBM teaches (p. 1) an AS/400 (Application System/400 with operating system OS/400, p. 400/data processing unit with memory) using an AFP printing and presentation system to transform digital print data streams, to communicate with printers through the Intelligent Print Data Stream (IPDS). The AS/400 (p. 28, interfaces stored in OS) and the printer devices (p. 38) both are data processing units with memory [a system for the transformation of digital print data streams comprising data processing unit having memory and communications interface, programmed to operate the method of claim 1]. IBM teaches (p. 1) system management to communicate via twinax or to be network-attached via TCP/IP [the communications interface]. See IBM, p. 405, "Systems Network Architecture" (SNA) and "TCP/IP." See additional limitations of claim 17 addressed in claim 1 rejection above.

Per **claim 21,** IBM teaches (p. 27, 264) LAN print servers that manage print job files to either IPDS or ASCII printers.

Re. Ground #2: **Claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, and 48-49 are rejected as obvious over IBM in view of Interleaf under 35 U.S.C. 103(a).** This rejection proposed by the Third Party requester in the request for inter partes reexamination at pages 5, 7-8 and claim chart (Exhibit H) is **adopted** as proposed.

IBM discloses the limitations of independent **claims 1, 17, 22, and 38** as noted above. To the extent that IBM fails to expressly teach "at least one stored script is assigned, which is executed" to the graphically representable objects, Interleaf discloses (p. 79) a "Lisp script...

script." Similar to the '789 patent, 6:6-16, Interleaf teaches (p. 84) event-based activation of scripts: "If a document contains active objects within the document file stream (as opposed to the methods file), these become activated when the document is opened programmatically or by the user for viewing, editing, or printing."

Regarding **claims 12, 33, and 49**, IBM teaches the graphically representable object stored in memory in an object oriented format is assigned at least one script. See teachings above. Interleaf further expressly teaches (78-79) assigning "at least one case relating to the execution of the script": A template document subclass is defined...has a custom open method that changes component names and sets other document properties according to the table-specified settings for the current language. This occurs while the document is opening, but before its content is shown. Defining a "custom open method" that runs "while the document is opening, but before its content is shown" shows defining at least one case relating to the execution of the script. Interleaf also teaches (p. 78-79) that the script is "occurring automatically, preferably without further influence from outside."

Regarding **claims 20 and 21**, as noted above, IBM teaches a printer / printer server characterized in that it has a system for the transformation of digital print data streams. Also see IBM definitions at p. 404.

Re. Ground #3: **Claims 1-14, 16-18, 20-35, 37-51 and 53 are rejected under 35 U.S.C. 103(a) as obvious over IBM in view of Interleaf, and further in view of the Interleaf Patent.** This rejection proposed by the Third Party requester in the request for inter partes reexamination at pages 10-12 and claim chart (Exhibit I) is **adopted** as modified. (It is presumed that the

alone or in combination with "Interleaf Patent" and "Lieberman", is considered to raise a substantial new question of patentability as to claims 1-2, 4-14, 15-19, 22-23, 25-39, and 41-53 of the '789 patent.

<u>**Re. Ground #5:**</u> **Claims 1-2, 4-7, 11-12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 are rejected under 35 USC 102(b) as being anticipated by Interleaf under 35 U.S.C. 102(b).** This rejection proposed by the Third Party requester in the request for inter partes reexamination at pages 10-14 and claim chart (Exhibit I) is **adopted** as proposed in the request.

Per **claims 1, 22, and 38**, Interleaf teaches (p. 75, 80) a document processing system that uses "active documents", which are documents "built with an extensible object system." (Interleaf, p. 75) and includes document system facilities for printing, filtering to accept CALS compliant [a government standard] documents [input data streams], and automatic hypertext linking and indexing." Interleaf describes (p. 84) receiving a "document file stream" for printing, and parses the document to see if it has any active objects: "If a document contains active objects within the document file stream ... these become activated when the document is opened programmatically [read in input data stream/ analyzed by means of a parser/ LISP interpreter @ p. 77] or by the user for ... viewing, editing, printing." Interleaf teaches (p. 76) changing the style "in the middle of a component text stream." [a method for transformation of a digital print data stream] Interleaf provides another example of a method for transformation of a digital print data stream (p. 81-84) by changing the objects in the stream or by programmatically apply different graphic properties or transformations to the object, such as changing its color or size or position. Interleaf teaches (p. 84) saving code of the document itself, either within the main document file stream or in an auxiliary file containing method definitions. [digital print data

Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to combine Interleaf and Lieberman for at least the following reasons: 1) The specific language used in scripting is "merely a matter of obvious engineering choice," and thus prima facie obvious. (MPEP § 2141.04.V.(B)). The '789 patent suggests no unexpected or unpredicted benefit of using Java Script in this manner, and in fact merely refers to Java Script as one of several "example" scripting languages that could be used. ('789 patent, 6:10-11). Thus, Lieberman teaches that substituting Lisp (as taught in Interleaf) for Java script is just substituting one known solution (Lisp) for another known solution (Java script). 2) A person of ordinary skill in the art would be motivated to combine Interleaf and Lieberman because they both address the same field, namely scripting interfaces for object systems. This overlap in the field of use would motivate a person of ordinary skill in the art to combine Interleaf and Lieberman.

<u>**Re. Ground #7:**</u> **Claims 1-2, 4-14, 16-18, 22-23, 25-35, 37-39, 41-51, and 53 are rejected as obvious over Interleaf in view of the Interleaf Patent under 35 U.S.C. 103(a).** This rejection proposed by the Third Party requester in the request for inter partes reexamination at pages 13-15 and claim chart (Exhibit K) is **adopted** as modified in the request. Examiner presumes a typo and has removed claim 15 from the list of rejected claims. The 35 USC 103(a) rejection of Claim 15 as obvious over Interleaf in view of the Interleaf Patent is **not adopted** as proposed under ground #7.

Regarding **claims 1, 17, 22, and 38**, Interleaf describes (p. 80, 84) a document processing system, active documents and an extensible objects system and to printing of such documents. Interleaf teaches (p. 76) a transformation (p. 81-82-transformations to the object) of the digital

in combination with the IBM reference is considered to raise a substantial new question of patentability as to claims 1-53 of the '789 patent.

<u>**Re. Ground #9:**</u> **Claims 1-53 are ejected under 35 USC 103(a) as obvious over the Interleaf Patent in view of IBM.** This rejection proposed by the Third Party requester in the request for inter partes reexamination at pages 7-8 and 16-17 and claim chart (Exhibit I) is **adopted** as proposed in the request. See Request p. 16-17 and Exhibit L.

The Interleaf Patent describes an "electronic document processing system" or EDPS that modifies documents for printing. Interleaf Patent at 1:8-11; 2:10-11. Fig. 5 of the patent, reproduced below, shows an example document in the form of a memorandum 32 having multiple objects, including a "graphic object 48." Interleaf Patent, 3:4. The Interleaf Patent describes providing scripts to the objects to trigger events during presentation or printing. A list of example events is provided in columns 6-9 of the patent. Interleaf Patent creates electronic documents for "printing."

IBM describes the Advanced Function Presentation and printing system (AFP). As noted above IBM teaches (See Request, 07/16/2010, p. 7) all the claim elements that were found to be missing in the original prosecution.

The teachings of Interleaf Patent are addressed in combination with IBM in more detail as noted above in the first SNQ.

In reference to Interleaf Patent, it is inherent that when printing, the objects of the active document to be printed are combined into an output stream. In the alternative, combining the

Attorney Docket No.: P-74637-US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patentee(s):	KRAUTTER, Thomas	Examiner:	STEELMAN, Mary J.
Control No.:	95/001,398	Reexamination Filed:	July 16, 2010
Patent No.	6,684,789	Art Unit:	3992
Title:	METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER		

AMENDMENT AND RESPONSE

Mail Stop Inter Partes Reexam
Attn: Central Reexamination Unit
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Amendment and Response is being filed by the patent owner CCP Systems AG ("CCP") in response to the Office action dated November 19, 2010 issued by the United States Patent and Trademark Office in connection with the above-identified re-examination, setting a one-month response date. A one-month extension of time has been granted, making the response due January 19, 2011. Accordingly, this Amendment and Response is being timely filed.

Amendments to the Claims are reflected in the listing of claims which begins on the following page.

Kindly consider the **Remarks** following the Claims Section of this paper.

AMENDMENTS TO THE CLAIMS

Please add or amend the claims to read as follows, and cancel without prejudice or disclaimer claims indicated as cancelled:

1. (Original) A method for the transformation of digital print data streams, in which
   (i) an input print data stream (2) is read in,
   (ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and
   (iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, and
   (iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and
   (v) the objects thus transformed are combined into an output print data stream (10) and are output,

   characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

2. (Original) The method as claimed in claim 1, characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

3. (Original) The method as claimed in claim 1, characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part

objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

4. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

5. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

6. (Original) The method as claimed in claim 5, characterized in that the script (5a) automatically receiving data also requests this data automatically.

7. (Original) The method as claimed in claim 5, characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

8. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

9. (Once Amended) The method as claimed in claim 8, characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

10. (Original) The method as claimed in claim 9, characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

11. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

12. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

13. (Original) The method as claimed in claim 12, characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

14. (Original) The method as claimed in claim 13, characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

15. (Original) The method as claimed in claim 1, characterized in that Java Script is used as a formal language for the scripts.

16. (Original) The method as claimed in claim 1, characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

17. (Original) A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates in accordance with the method as claimed claim 1.

18. (Original) The system for the transformation of digital print data streams as claimed in claim 17, the system also has an operating station with display means (8) and input means, which makes it possible for the graphically representable objects (5) stored in the memory (6) of the data processing unit in an object-oriented format, preferably also script objects (5a), to be read out via the application interface (7), to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream (10).

19. (Original) The system for the transformation of digital print data streams as claimed in claim 17, wherein the data processing unit, permits respectively stored objects, preferably also Java Script objects (5a) themselves, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations if necessary being transformed automatically into Java Script objects (5a).

20. (Original) A printer, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.

21. (Original) A printer server, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.

22. (Original) A computer-readable medium having stored thereon instructions to cause a processor to execute a method, the method comprising: (i) an input print data stream (2) is read in, (ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and (iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, (iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and (v) the objects thus transformed are combined into an output print data stream (10) and are output, characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

23. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

24. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

25. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

26. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

27. (Original) The computer-readable medium as claimed in claim 26, the method characterized in that the script (5a) automatically receiving data also requests this data automatically.

28. (Original) The computer-readable medium as claimed in claim 26, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigend by itself.

29. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

30. (Original) The computer-readable medium as claimed in claim 29, the method characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

31. (Original) The computer-readable medium as claimed in claim 30, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and forwards the graphic object (5) associated with itself to a receiver together with the data requested, received and reassigned by itself.

32. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

33. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

34. (Original) The computer-readable medium as claimed in claim 33, the method characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

35. (Original) The computer-readable medium as claimed in claim 34, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

36. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that Java Script is used as a formal language for the scripts.

37. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

38. (Original) A computer data signal embodied in a carrier wave and representing sequences of instructions which, when executed by a processor, cause the processor to perform a method, the method comprising: (i) an input print data stream (2) is read in, (ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and (iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, (iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and (v) the objects thus transformed are combined into an output print data stream (10) and are output, characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

39. (Original) The computer data signal as claimed in claim 38, the method characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

40. (Original) The computer data signal as claimed in claim 38, the method characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

41. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

42. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

43. (Original) The computer data signal as claimed in claim 42, the method characterized in that the script (5a) automatically receiving data also requests this data automatically.

44. (Original) The computer data signal as claimed in claim 42, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

45. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

46. (Original) The computer data signal as claimed in claim 45, the method characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

47. (Original) The computer data signal as claimed in claim 46, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and forwards the graphic object (5) associated with itself to a receiver together with the data requested, received and reassigned by itself.

48. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

49. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

50. (Original) The computer data signal as claimed in claim 49, the method characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

51. (Original) The computer data signal as claimed in claim 50, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

52. (Original) The computer data signal as claimed in claim 38, the method characterized in that Java Script is used as a formal language for the scripts.

53. (Original) The computer data signal as claimed in claim 38, the method characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented

format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

54. (**New**) A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates to perform the following:

(i) read an input print data stream;

(ii) analyze the input print data steam by means of a parser for graphically representable objects and split up the input print data stream into these graphically representable objects;

(iii) store said graphically representable objects in a memory in an object-oriented format;

(iv) transform the graphically representable objects into a format for the control of a printer; and

(v) combine the objects into an output print data stream, and output said combined output print data stream to said printer,

wherein said graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

55. (**New**) The system according to claim 54, characterized in that the data processing unit is further programmed to send and receive e-mails in the cases defined in the script.

56. (**New**) The system according to claim 54, characterized in that the data processing unit is further programmed to print original print and image data without a printer driver.

57. (**New**) The system according to claim 54, further comprising:

an operating station with display means and input means, wherein said operating system makes it possible for the graphically representable objects to be read out via the application interface, to be changed, to be deleted, or to be appended before they are output in the output print data stream.

58. (**New**) The system according to claim 57, wherein said graphically representable objects to be read out, changed, deleted, or appended via the application interface are script objects.

59. (**New**) A printer adapted for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer is adapted to transform the graphically representable objects into a format for the control of an output device, to combine the objects into an output print data stream, and to output said combined output print data stream, and

wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in the cases defined in the script.

60. (**New**) The printer of claim 59, wherein said script is configured to operate as a timer, such that the script is executed automatically as a result of expiry of time.

61. (**New**) The printer of claim 59, wherein said script when executed operates to obtain information over a network, and to use the obtained information in the output print data stream.

62. (**New**) The printer according to claim 59, wherein said printer is adapted to print original print and image data without a printer driver.

63. (**New**) The printer according to claim 59, wherein said printer is further adapted to send and receive e-mails in the cases defined in the script.

64. (**New**) A printing system comprising

a printer adapted for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer is adapted to transform the graphically representable objects for the control of an output device, to combine the objects into an output print data stream, and to output said combined output print data stream, and

wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in the cases defined in the script; and

an operating station with display means and input means, wherein said operating station is adapted to allow manipulation of said at least one script via an application interface.

65. (**New**) The printing system according to claim 64, wherein said manipulation comprises at least one operation selected from the operations consisting of: reading out, changing, deleting, or appending said script.

66. (**New**) The printing system according to claim 64, wherein said printer is adapted to print original print and image data without a printer driver.

67. (**New**) The printing system according to claim 64, wherein said printer is further adapted to send and receive e-mails in the cases defined in the script.

68. (**New**) A printer server for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer server is adapted to transform the graphically representable objects for the control of an output device, to combine the objects into an output print data stream, and to output said combined output print data stream to the output device, and

wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in the cases defined in the script.

69. (**New**) The printer server of claim 68, wherein said script is configured to operate as a timer, such that the script is executed automatically as a result of expiry of time.

70. (**New**) The printer server of claim 68, wherein said script when executed operates to obtain information over a network, and to use the obtained information in the output print data stream.

71. (**New**) The printer server of claim 68, wherein said output device is a printer.

72. (**New**) A printing system comprising

a printer server adapted for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer server is adapted to transform the graphically representable objects into a format for the control of a printer, to combine the objects into an output print data stream, and to output said combined output print data stream to the printer, and

wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in the cases defined in the script; and

an operating station with display means and input means, wherein said operating station is adapted to allow manipulation of said at least one script via an application interface.

73. (**New**) The printing system according to claim 72, wherein said manipulation comprises at least one operation selected from the operations consisting of: reading out, changing, deleting, or appending, said script.

74. (**New**) The printing system according to claim 72, wherein said printer server is adapted to print original print and image data on the printer without a printer driver associated with the printer.

75. (**New**) The system according to claim 72, wherein said printer server is further adapted to send and receive e-mails in the cases defined in the script.

76. (**New**) The method of claim 1, wherein said input data stream is formatted in a page description language.

77. (**New**) The method of claim 76, wherein said parser is a syntax analyzer, and wherein said analyzing by means of said parser comprises performing syntactic analysis on said input data stream.

78. (**New**) The method of claim 1, wherein storing the graphically representable objects in said object-oriented format comprising storing said graphically representable objects based on membership in hierarchically organized classes.

79. (**New**) The method of claim 1, further comprising dynamically linking a plurality of objects.

80. (**New**) The method of claim 1, wherein the objects are managed by display list management module.

81. (**New**) The method of claim 80, wherein the display list management supports one page and multi-page documents at a plurality of levels.

82. (**New**) The method of claim 81, wherein the display list management can be expanded dynamically by new objects.

REMARKS

This *inter partes* reexamination relates to US Patent No. 6,684,789 ("the '789 Patent"). The Examiner has issued an office action holding claims 1-53 invalid in view of several prior art references discussed below.

This response and the accompanying declarations and exhibits are intended to be fully responsive to all points of objection and/or rejection raised by the Examiner and are believed to place the claims under reexamination in condition for confirmation. The following declarations are provided together with this response:

- Declaration of Roland Widuch, Chief Executive Officer of CCP ("Widuch Dec.")
- Declaration of Christoph Picht, Director of Business Development at CCP ("Picht Dec.")
- Declaration of David Birnbaum, Ph.D. ("Birnbaum Dec.")

Favorable reconsideration and confirmation of all claims under reexamination is respectfully requested. As explained below, the claimed inventions are new, non-obvious and useful. Indeed, based on the years of commercial exploitation of the patented subject matter by both the Requester, Samsung Electronics Corp., Ltd. ("Samsung") and IBM Deutschland GmbH ("IBM"), the patented technology is certainly innovative and would not have been obvious. Prompt consideration and confirmation of the claims is respectfully requested.

Status of Claims

Claims **1-53** are subject to reexamination. Claims **1-53** have been rejected.

Claims **54-82** have been newly added in this response. The claims do not enlarge the scope of the patent or introduce new matter. Support for the new claims is provided at Appendix A hereto.

Status of Litigation Proceeding

In addition to the pending reexamination proceedings, patentee, CCP Systems AG ("CCP") is plaintiff in an ongoing litigation *CCP Systems AG v. Samsung Electronics Corp., Ltd. Samsung Electronics America, Inc., Samsung Networks, Inc. and IBM Corp.*, 2:09-cv-04254 (D. N.J.). CPP has accused various Samsung entities of copyright and patent infringement. CCP has also asserted a copyright infringement claim against IBM.

The Court granted Samsung's and its subsidiary Samsung America, Inc's request to stay the patent infringement claims asserted against them, but denied their request to stay related copyright infringement claims asserted against Samsung and its subsidiaries.

In the litigation, Samsung has <u>admitted</u> "that firmware for certain SEC [Samsung Electronics Corp. Ltd., requestor here] printers has included JScribe Core software or software adapted from JScribe Core software." (Complaint ¶ 17, attached as Appendix B). As the '789 patent explains, JScribe®, CCP's copyrighted and trademarked software, is covered by the patent in this reexamination (see '789 patent, Col. 5, lines 55-59). (Picht Dec. ¶ 11-21; see also Complaint ¶¶ 31, 52-54, 72 and 81 Appendix B).

There has been no discovery in the litigation, so CCP does not know the precise extent of Samsung's infringement. But, Samsung's size is well known, and CCP believes Samsung sold at least half a million infringing printers in the United States, and many more than that on a worldwide basis. (Widuch Dec. ¶ 5). Samsung also allowed its customer and partners to download CCP's JScribe® technology from Samsung's website without permission from CCP. (Widuch Dec. ¶ 6). Again, CCP does not know the full extent of Samsung's infringement, but believes there was a large number of downloads as well.[1] (Widuch Dec. ¶ 6).

We mention this here, and discuss the significance in more detail below, because IBM's and Samsung's extensive, worldwide exploitation of the technology at issue – and IBM's substantial payments for the right to license the technology – seriously undermine Samsung's

---

1 Samsung can provide the exact numbers of printers it sold that contain JScribe® technology as well as the amount of its license payments to IBM in its response, if it chooses.

claim in this reexamination that the technology was allegedly old, and had been used by IBM before the filing date. Obviously, if IBM already had the technology, as Samsung now claims, IBM would never have paid to license it.

FACTUAL BACKGROUND OF THE REEXAMINATION

It is critically important for the Examiner to understand the historical background of this matter because it demonstrates that the Requester Samsung's current contentions are diametrically inconsistent with its pre-litigation actions.

The IBM License

In 2003-2004 IBM Germany evaluated CCP's technology, including the patent at issue here. (Widuch Dec. ¶ 7). After satisfying itself of the value of this technology (and inherently its innovativeness), in 2004 IBM's German subsidiary took an exclusive license to CCP's JScribe® technology ("the IBM License"). (Widuch Dec. ¶ 7). The exact amounts may be confidential to IBM and CCP will not disclose them here. However, the agreement committed IBM to pay a minimum of tens of millions of dollars in royalties over the next five years. (Widuch Dec. ¶ 7). In 2005, IBM was given the right to sub-license the technology under certain conditions. (Widuch Dec. ¶ 7).

IBM's first sub-licensee under the agreement, Konica-Minolta Business Solutions, Inc., ("Konica-Minolta") paid IBM millions of dollars in a lump-sum payment plus agreed to a running royalties for the right to imbed CCP's technology into its products. (Widuch Dec. ¶ 7).

In February 2005, IBM and CCP entered into an agreement called a "Co-Marketing Logo License Agreement." (Widuch Dec. ¶ 9). That agreement set out the parties' rights vis-à-vis trademarks used in certain marketing material. It also included examples of acceptable marketing materials, an example of which is at Widuch Exhibit A. That example identifies the product as "JScribe® Print Server Solution 1.0". Further, is says the product was "Developed by CCP Systems AG", and "This program is protected by copyright laws and U.S. patent 6,684,789 [the patent at issue here]." IBM's name is also prominently displayed on the advertising copy.

Among other things, this is a recognition by IBM that JScribe® is covered by the patent at issue, and also evidence that IBM regarded the '789 patent as valid.

The Samsung Sub-License

In about 2006, IBM, through its subsidiary in Korea, sub-licensed the technology to Samsung. (Widuch Dec. ¶ 10). Samsung paid IBM a lump sum in the millions of dollars, and agreed to pay a running royalty for each device that included the licensed JScribe® technology. (Widuch Dec. ¶ 10).

Although JScribe® was at all times CCP's property, IBM and Samsung have at various times claimed it as their own. For example, a joint IBM-Samsung press release stated: "'JScribe', the printing optimized middleware of IBM will be launched in the Samsung digital printing devices such as printers and multifunction products." (Emphasis added; Widuch Ex. B).

Similarly, in about 2008, Samsung issued a press release entitled: "New Samsung MFP [multi function printer] with Advanced Features Offers Easy Integration for Efficient, Simple and Reliable Printing" that touted advantages of CCP's product that Samsung characterized as "*Samsung's* JScribe™". (Widuch Dec. Ex. E, p. 2; emphasis added).

JScribe® software has won various innovation awards. In 2006, IBM awarded CCP its "Best Seller Award," a recognition of CCP's groundbreaking work on JScribe® technology. (Widuch Dec. Ex. C). In June 2007, CCP's JScribe® product won the European Commission China Information and Communications Technologies Innovator Award (ChinICT). (Widuch Dec. Ex. D). CCP was selected from 60 enterprises as among the best innovators. (Widuch Dec. Ex. D).

Termination of the IBM and Samsung Relationships

In January 2007, IBM sold the remainder of its printer business to Ricoh Company, Ltd. and terminated the exclusivity of the IBM License. (Widuch Dec. ¶ 13). IBM continued to pay CCP some, but not all, of the royalties that IBM was obligated to pay under the IBM License as a result of Samsung's sales. (Widuch Dec. ¶ 13). CCP currently has litigation pending in

Germany against IBM related to the amount of royalties that IBM was supposed to pay under the IBM License. (Widuch Dec. ¶ 13).

In 2009, CCP terminated the License with IBM, and with it any rights that Samsung may have had under that agreement. (Widuch Dec. ¶ 19). Interestingly, even after having been sued, and after bringing on this reexamination, Samsung's web site continues to promote the advantages of "JScribe technology." (Widuch Ex. G).

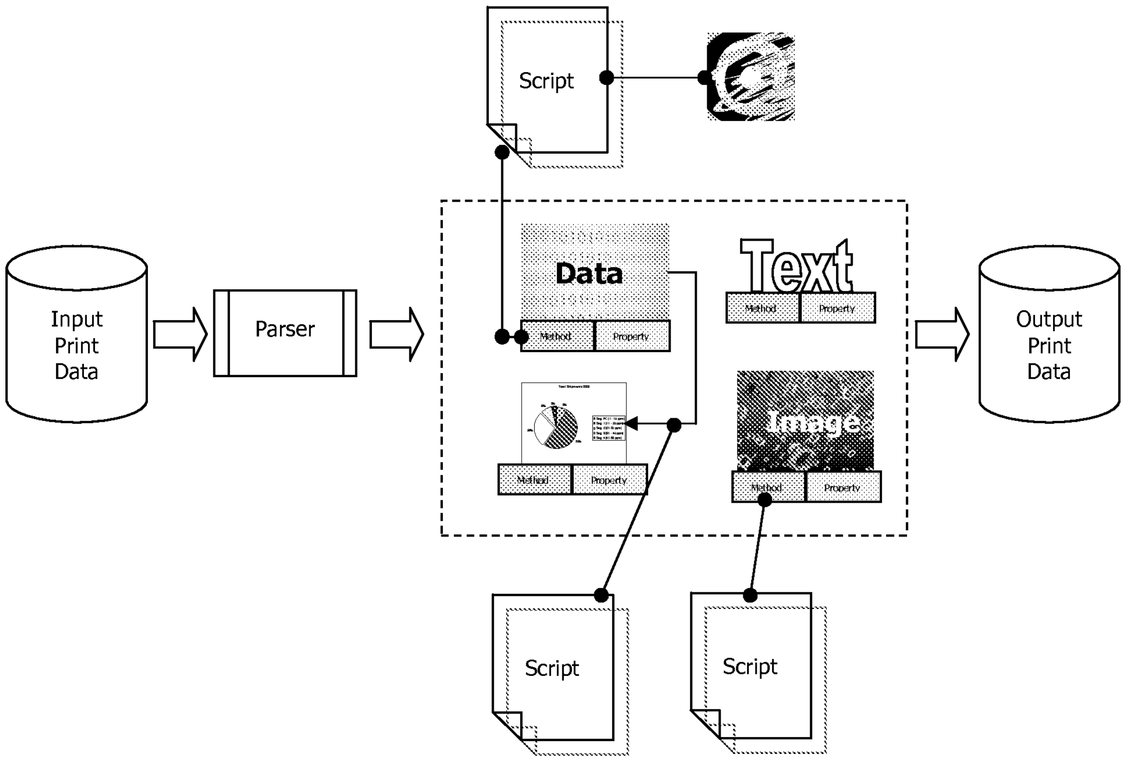
We discuss these historical facts in more detail below, but they are exceptionally strong proof of *objective* evidence that the '789 patent claims, which cover CCP's JScribe® product (Picht Dec. ¶ ¶ 11-21), would not have been obvious to those of ordinary skill. Had it been obvious, neither IBM nor Samsung would have paid millions of dollars to license the technology from CCP.

In addition to the '789 patent, CCP obtained corresponding patents in Europe (EP 1282883 B1) and Japan (JP 3974782 B2). (Picht Dec. ¶ 22).

CLAIM REJECTIONS

Introduction to the Technology

The '789 patent claims cover a highly successful software product called JScribe®, sold by the patentee, CCP Systems AG. (Picht Dec. ¶¶ 11-21; see also Widuch Dec. ¶¶ 7-18 and 20). The software, which typically resides in printers or multi function devices, receives an input print stream and parses the print steam into objects, which are then stored in object-oriented format. (See generally, '789 patent Col. 8, lines 55-65). At least one script is assigned to an object, which is executed in the cases defined by the script. Upon being sent for printing, the objects are transformed into a format for the control of a printer, and the transformed objects are combined into an output print data stream. This process is depicted schematically below:



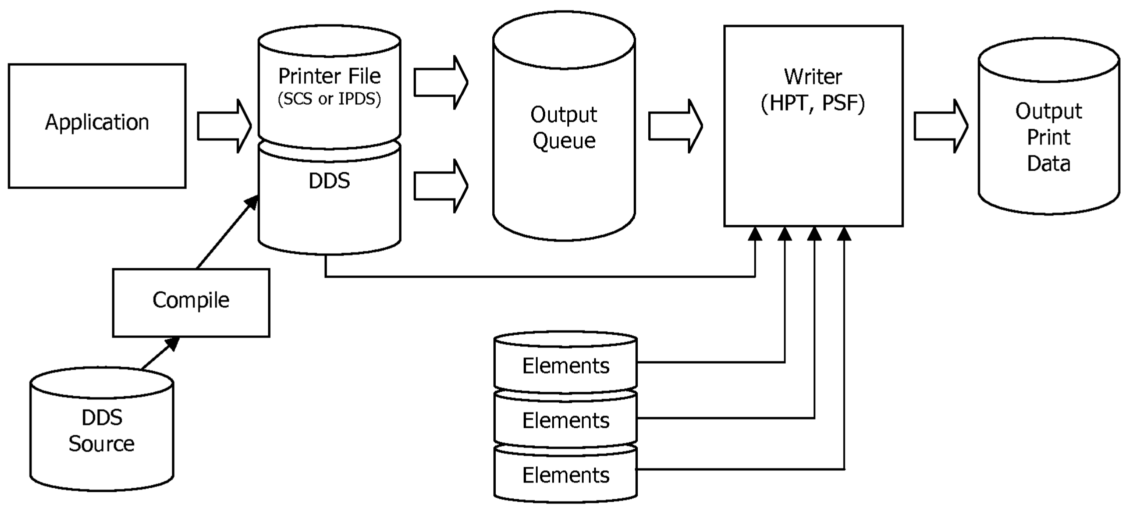
By parsing the print stream into object-oriented format, as described and claimed in the '789 patent, the system attains enormous flexibility, such as allowing real-time updates (e.g., stock prices over the Web) at printing time (see '789 patent Col. 6, lines 40-41) and allows for robust two-way communication with the printer. Moreover, by using scripts, the patented software is output-device independent, allowing an organization with multiple output devices in a heterogeneous environment to control them optimally in a distributed fashion. (See '789 patent, Col. 2, line 62 – Col. 3, line 4).

As discussed below, the cited references – IBM, Interleaf, Interleaf Patent, and Lieberman – do not include every claim element. Nor would it have been obvious to one of ordinary skill in the art to combine these references to thereby arrive at the claimed invention. Even if the references had somehow been combined, the result would not have been any of the claimed inventions.

Brief Summary of References

1. IBM AS/400 Guide to Advanced Function Presentation and Print Services Facility ("the IBM Reference"):

IBM's AS/400 is a midrange computer system that supports multiple users. The reference describes Advanced Function Presentation ("AFP"). AFP allows graphical presentation of data produced by applications running on the AS/400. The AS/400 receives data from an application (including text, images, etc.), prepares the data for printing (including, for example, by executing a DDS (Data Description Specification) on the data) as output print data, and sends this output data to the selected printer. (Birnbaum Dec. ¶ 8).



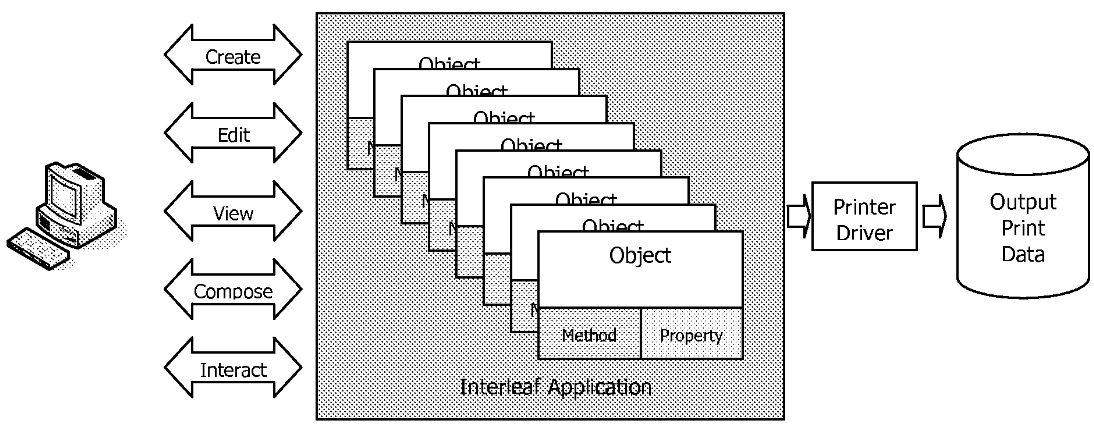
By picking and choosing among the 400 pages of the AFP guide, the Requester has purported to find all claim elements, but the similarities are superficial, at best. Closer inspection of the IBM reference reveals that essential components of claims are entirely missing. As described below, the IBM reference fails to disclose at least:

- parsing the input data stream;
- storing graphically representable objects in objected-oriented format;

- using scripts; and
- assigning at least one script to an object.

2. *Interleaf and Interleaf Patent*

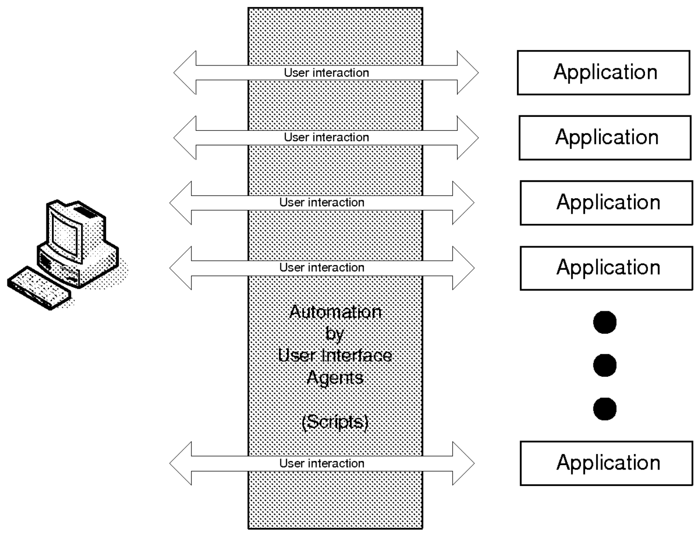
Interleaf Active Documents ("Interleaf") and U.S. Patent No. 5,579,519 ("Interleaf Patent") describe creation of a so-called "active documents" (together referred to as the "Interleaf documents"). However, "active documents" are created at the computer itself, far upstream from the data discussed in the IBM reference (or the '789 patent) (see Figure below). Documents using the Interleaf disclosures may certainly be printed; however, the only print stream is the output of a print process, and the treatment of this print stream is not discussed in any detail in the Interleaf reference. (Birnbaum Dec. ¶ 9). The technology in the '789 patent, by contrast, covers methodology that *prints* documents, *after* they have been created by other programs; that is, it has a print data stream as its input, which is then transformed through the patented technique into an output data stream. This process of transforming a print data stream is not disclosed or suggested in the Interleaf documents.



3. *Lieberman*

The article Integrating User Interface Agents with Conventional Applications, by Henry

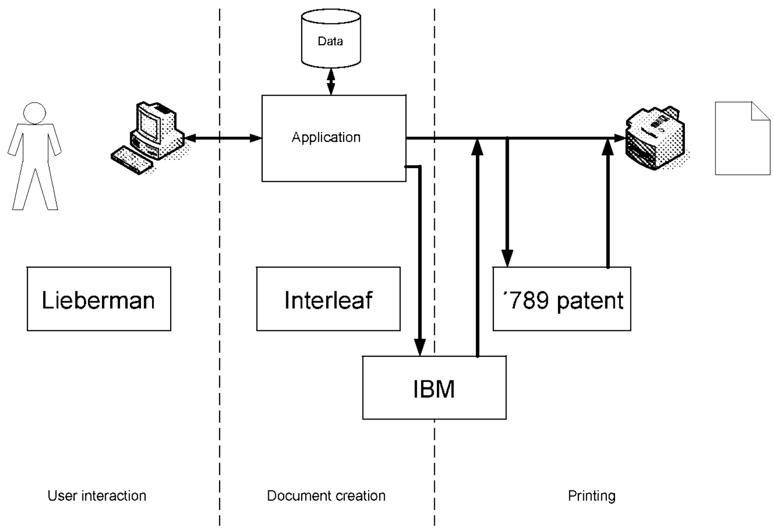
Lieberman ("the Lieberman reference" or "Lieberman"), describes automating user interaction by using scripts. (Birnbaum Dec. ¶ 10) (see Figure below). Lieberman is really far afield from the patented technology. Lieberman is not related to printing at all, and cannot be considered the same field as the patent at issue. Therefore, combining these disparate references is not



something a person of ordinary skill would have done.

* * *

The following schematic diagram shows the location in the document creation and printing "food chain" where these various references may be found:



(Birnbaum Dec. ¶ 11).

As discussed in further detail below, neither the IBM reference nor the Interleaf documents include every element of the '789 patent claims. Moreover, the IBM, Interleaf, and Lieberman are each at different stages of the print workflow. Only with improper hindsight – and complete system redesigns – would it have been "obvious" to "cherry-pick" features of each of these references to create the inventions of the '789 patent.

I. Ground of Rejection No. 1: IBM

The Examiner adopted the Request's proposed rejection of claims 1-5, 17, 20-26, and 38-41 under 35 U.S.C. § 102(b) as anticipated by the IBM reference. See Request pp. 7-8, 10-12, and Exhibit G. The rejection is traversed for at least the following reasons.

A. IBM does not disclose a parser to analyze a print data stream

The claims recite analyzing an input print data stream for graphically representable objects by means of a parser. Contrary to the Request and the Office action, IBM does not disclose a parser as defined in the '789 patent.

A "parser" must be understood as that term is used and defined in the '789 patent:

> In this case, as opposed to the use of a status machine, the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used. (Col. 3, lines 21-26, emphasis added)[2]
>
> * * *
>
> [A] parser is used for the analysis and splitting up into the graphically representable objects . . . which is therefore capable of analyzing and splitting up languages with "context-free grammars". . . (Col. 4, lines 32-38, emphasis added)

A parser as properly understood according to the '789 patent, therefore, is a syntax analyzer: it is capable of analyzing the syntax of an input data stream based on a formal grammar. (Birnbaum Dec. ¶ 12).

The term "parser" in the '789 patent is consistent with a well-accepted meaning of the phrase in computer science. This functionality permits the parser to analyze even complex page description languages (PDLs), in comparison to a state machine driven by regular expressions, which (as shown below in reference to the IBM reference) can only reasonably handle single commands in a line by line approach. (Birnbaum Dec. ¶ 13).

Samsung purports to find a parser at IBM reference, pp. 194-195, including:

---

[2] The term "status machine" used in the '789 patent is likely an incorrect translation from the original German and should be understood as a "state machine."

> Specifically, the page definition defines how data is placed on a logical page layout. Input print lines are read in, optionally parsed into individual fields, and place [sic] on the page. (IBM, p. 194).

However, the function described in the IBM reference is capable of being performed by a state machine, and would not require a parser, as described and claimed in the '789 patent. (Birnbaum Dec. ¶ 14). This difference would have been understood by those of ordinary skill in computer science generally, and in the field of the invention in particular. Specifically, as discussed below, by referring to the data fields described at IBM reference, pp. 14-15, it is clear that input print lines are composed of structured fields. Identifying and separating the individual fields from a structured field is a simpler operation than that required for parsing, and can be performed by a state machine.[3] (Birnbaum Dec. ¶ 15).

The input process described in the IBM reference involves receiving lines in SCS (SNA [System Network Architecture] Character Stream) or IPDS (Intelligent Print Data Stream) format (defined below), and converting data fields in such lines into respective fields in AFPDS (Advanced Function Print Data Stream). (Birnbaum Dec. ¶ 16).

Regarding SCS, the IBM reference explains:

> The SNA character string (SCS) data stream has a relatively simple structure, consisting of a one-byte hexadecimal code followed by the data to be printed. SCS, which is the standard pre-AFP print data stream, is used to control line printers and supports row and column functions. (IBM reference, p. 14, emphasis added)

An Intelligent Printer Data Stream (IPDS) print data stream is structured according to the following format:

The IPDS Command Format

All IPDS commands are encoded in the following format:

Length	Command	Flag	CID	Data

[3] In the same document, IBM states that "the print line is then parsed (subdefined into individual fields) to define the zip code field" (p. 215, emphasis added), further demonstrating the IBM reference's loose use of the term "parsing" unrelated to syntactic analysis.

(Birnbaum Dec. ¶ 17; IBM iSeries Printer Device Programming, Version 5, Document No. SC41-5713-05, published September 2002, p. 492, relevant excerpt attached as Exhibit B to Birnbaum Dec.). In particular, "[t]he structured field format of IPDS allows commands to be sent to the printer in a continuous stream. Each command is self-describing." (Birnbaum Dec. ¶ 18; IBM SC41-5713-05, Ex. B to Birnbaum Dec.). That is, IPDS has no grammar and no syntax across commands. (Birnbaum Dec. ¶ 18).

It is in this context of the structured format of SCS and IPDS that the cited statement in the IBM reference ("Input print lines are read in, optionally parsed into individual fields, and place [sic] on the page"), which Samsung relies on, must be understood. These "input print lines" are made up of structured data fields, as explained at IBM reference, pp. 14-15. This operation of identifying and separating the individual data fields in a structured format is a different and simpler operation than parsing. (Birnbaum Dec. ¶ 19).

The result of the conversion is data in AFP data structure (AFPDS) format, which is another structured field format, shown in IBM reference Fig. 3, at p. 21, reproduced below:

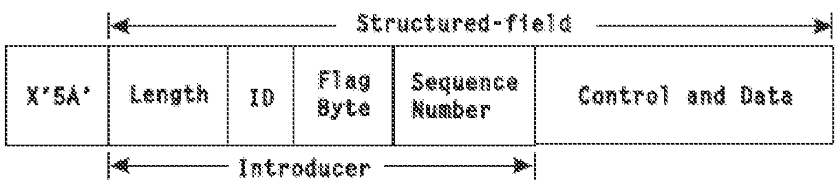
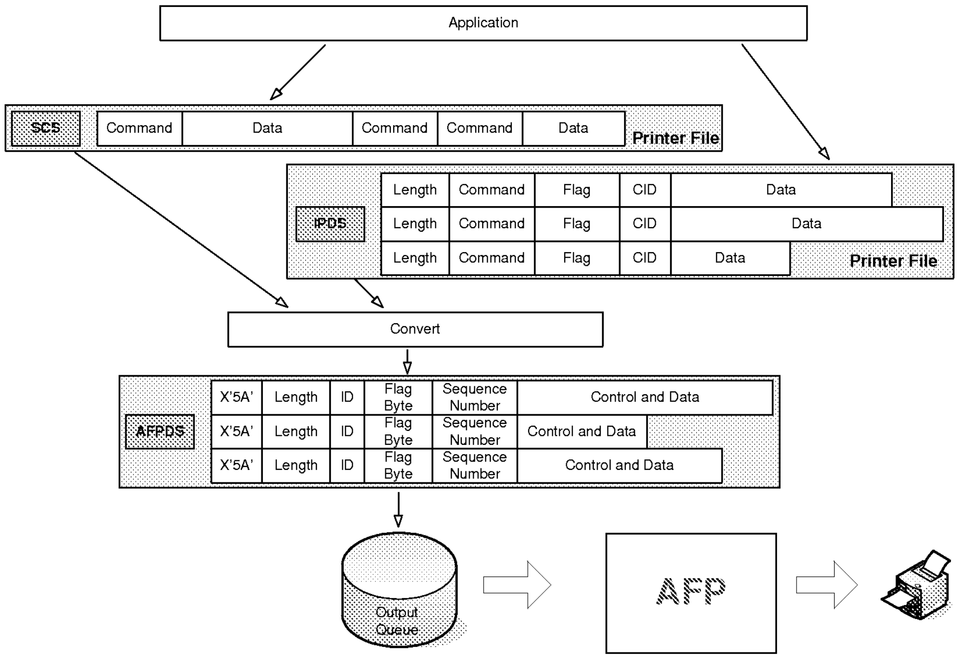


Figure 3. Structured Field

(Birnbaum Dec. ¶ 20). A field in AFPDS is identified by a leading 0X5A byte, and is followed by a number of fields that identify the length of the data, the type of data "ID", a flag byte and a sequence number, followed by the data. (Birnbaum Dec. ¶ 21).

The above input process, including the structured formats of the input SCS and IPDS data streams, as well as the resulting AFPDS format, may be understood by reference to the below figure.



The critical distinction is this: **the location and content of the structured fields are predefined and do not require parsing, since their location is already defined, as is the meaning of the values in each field**. (Birnbaum Dec. ¶ 21). Thus, evaluation and processing of an input print line in the IBM reference, which is defined as a structured field, can be done by a series of if-then statements. (Id.) Such an evaluation is one implementation of a state machine, which the '789 patent expressly distinguishes from the parser used in the invention. (See '789 patent, Col. 3, lines 21-30; Birnbaum Dec. ¶ 22).

In contrast, as explained above, parsing, as used in the '789 patent, is a far more complex operation; it requires scanning the input data stream character by character, identifying tokens, e.g., collections of characters that correspond to commands or data and a syntax evaluation to determine the relationships between the identified tokens. (Birnbaum Dec. ¶ 23). Often there is no rigid format for the input data stream. (Id.)

The IBM reference therefore does not disclose a parser as claimed in the '789 patent. Rather, the IBM reference only discloses conversion from structured SCS or IPDS format to

AFPDS. That function is not parsing within the meaning of the patent claims, because it does not require syntax analysis. (Birnbaum Dec. ¶ 24). Therefore, for this reason alone, the IBM reference did not anticipate the '789 patent claims.

B. IBM does not disclose storing graphically representable objects in object-oriented format

The claims recite storing graphically representable objects in object-oriented format. This too, contrary to the Request and the Office action, is not disclosed in the IBM reference.

In computer science, "object-oriented programming" uses objects – i.e., data, properties and methods, together with their interactions as defined by the methods. (Birnbaum Dec. ¶ 25). Object-oriented programming techniques typically include features such as class hierarchy, data abstraction, encapsulation, modularity, polymorphism, and inheritance. (Birnbaum Dec. ¶ 26).

Indeed, the '789 patent expressly discusses object-oriented features such as object class hierarchy and inheritance:

> The individual graphic objects are stored by using their membership of specific--expediently suitably hierarchically organized—classes. . . For example, via an object of the type square, it is already known from the object hierarchy that this is a subclass of the rectangle.
>
> [I]mplicit information. . . can be derived from the object class hierarchy. . . automatically also being available to the objects of subordinate, lower-ranking classes by way of inheritance. . . (Col. 3, line 38 – Col. 4, line 8, emphasis added)

An object-oriented format in the sense of the '789 patent, and as generally understood in computer programming, requires that objects have the ability to be organized, for example, in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend. (Birnbaum Dec. ¶ 27).

Some advantages of using object class hierarchy are described in detail in the '789 patent:

> The object-oriented . . . format . . . makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended. . . by

> assigning the methods required for this to the respective objects in accordance with their class hierarchy. (Col. 7, lines 24-30, emphasis added)
>
> * * *
>
> [F]eedback messages referring to the output print data stream output are read in and are analyzed for error messages which indicate that the output device. . . has recognized a transformed graphic object in the output print data stream which cannot be output by said printer, <u>this graphic object is then split up into part objects of lower complexity</u>. . . (Col. 4, lines 40 – 46).

If, for example, the printer is not capable of recognizing and outputting a bar-code object "then the bar code is simply split up into objects of the next lower hierarchy and a further try is made with these objects." This is continued if necessary until the printing is successful. The object-oriented data structure with its object hierarchy, chosen for the intermediate format, also proves to be particularly suitable for this procedure." (Col. 4, line 39 – Col. 5, line 2).

Samsung purports to find "object-oriented architecture" at IBM reference, pp. 21, p. 23, Fig. 5. However, the IBM reference uses the term "object" in a wide variety of ways, as exemplified by the thirteen definitions of the word in IBM's IBM Dictionary of Computing, 10th Ed., 1993). The IBM dictionary separately defines an "object" in the context of object-oriented design as distinguished from its use in connection with the AS/400:

> object. . . (10) In object-oriented design or programming, an abstraction consisting of data and the operations associated with that data. . . (11) In the AS/400 system, a named storage space that consists of a set of characteristics that define itself and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed; for example, programs, files, libraries, and folders. (IBM Dictionary of Computing, 10th Ed., 1993, relevant excerpt copied at Appendix C).

The IBM reference uses the term "object" to refer to data structures in which various document elements (e.g., image, form, text, font, bar code, and graphic) are represented. For a number of reasons -- including lack of: class hierarchy, assignment of methods to objects and inheritance -- "object," as used in the IBM reference, does not have the same meaning as "object" in object-oriented programming, and as that term is used in the '789 patent. (Birnbaum

Dec. ¶ 28). Indeed, the structured field format of AFPDS may be suitable for storing such data elements, but it is not suitable for storing objects in object-oriented format. (Id.)

So, for example, page 23 of the IBM reference, purports to describe as an "object" an image of an airplane. However, that so-called "object" is not handled as a member of a class to which dedicated methods are assigned – and therefore cannot be considered an object in the object-oriented programming sense. (Birnbaum Dec. ¶ 29).

Moreover, the IBM reference uses Mixed Object: Document Content Architecture-Presentation (MO:DCA-P) (IBM, p. 21).

> IBM has defined a single object-oriented data stream—Mixed Object Document Content Architecture (MO:DCA). (An object is a collection of data that can be treated as a unit.) (Emphasis added).

See, IBM iSeries Printer Device Programming, Version 5, Document No. SC41-5713-05, published September 2002, p. 495, relevant excerpt at Birnbaum Dec. Ex. B.

The IBM reference discloses simple data structures whose data are treated as a unit – and nothing more. (Birnbaum Dec. ¶ 30). By contrast, an object-oriented format, in the sense of the '789 patent and elsewhere, requires that objects be capable of being organized in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend. (Birnbaum Dec. ¶ 31).

When all the evidence is considered, the IBM reference does not disclose graphically represented objects stored in object-oriented format, a limitation used in all the claims of the '789 patent.

C. IBM does not disclose use of a script

The claims recite that a stored script is assigned to a graphically representable object. The Requester has pointed to the IBM reference and claimed that the Data Description Specification ("DDS") is a script. However, this is based on a fundamental misunderstanding of the meaning and importance of a script (and of how DDS operates).

A script – as distinguished from other types of programming languages is a program or series of commands that is interpreted and runs in real time rather than compiled and then executed. (Birnbaum Dec. ¶ 32). One definition of a script is a "series of commands in a high-level programming, or command, language that is interpreted (translated on the fly) rather than compiled ahead of time." (Birnbaum Dec. ¶ 33). In contrast, programs written in standard program languages, such as FORTRAN, C++, etc., must typically be compiled into object code before being executed. (Birnbaum Dec. ¶ 34).

The above distinction is important to attain the benefits of the claimed inventions. In particular, using scripts provides at least two advantages: flexibility and portability. (Birnbaum Dec. ¶ 35).

First, one feature of scripts is that they may be freely modified, including at run-time, which is not trued for compiled code. For example, in one embodiment of the invention, the data processing unit permits stored objects, including particularly script objects, to be read out graphically, to be changed, to be deleted or to be appended. See, e.g., claims 18, 19. This can only be done using a script, not compiled object-code. (Birnbaum Dec. ¶ 36).

Second, object code is limited because it is platform dependent. That is, a program written in a compiled language must be compiled into object code for a particular processor architecture. In contrast, any processor capable of interpreting a script can run the script. This renders scripts portable, i.e., platform-independent. (Birnbaum Dec. ¶ 37).

The Request purports to find a script in IBM's use of Data Description Specification (DDS). However, the IBM reference does not use the word "script", nor does it refer to DDS as "scripts"; on the contrary, as discussed below, DDS must be compiled before being used:

> The DDS will be **compiled** before the application program runs. The application program never looks at the DDS file and member, only at the **compiled** results. . . You can update the DDS; however, you must then re-compile it. (IBM iSeries Printer Device Programming, Version 5, Document No. SC41-5713-05, published September 2002, p. 61, emphasis added, relevant excerpt copied at Birnbaum Dec., Ex. B).

That is, the DDS is compiled some time before being used for printing, and can only be executed as object code after being compiled. Therefore, the compiling of a DDS is independent and asynchronous to using the DDS. (Birnbaum Dec. ¶ 38).

This feature of DDS makes it different (by definition) from a script, in which interpretation and execution of the script are typically simultaneously triggered, in the case of the '789 patent, for example, by the incoming print data. (Birnbaum Dec. ¶ 39).

Therefore, contrary to Samsung's arguments, the DDS described in the IBM reference are not scripts as claimed in the '789 patent, nor are they even referred to in the IBM reference as such. This critical difference is a fundamental distinction between the IBM reference and the patented subject matter.

D. IBM does not disclose a script assigned to an object

All claims require that at least one script (as defined above) be assigned to an object (stored in an object-oriented format). As discussed above, the "elements" in the IBM reference (images, text, etc.) are not objects stored in object-oriented format. Moreover, the DDS used in IBM is not a script. Even if an "element" were an object, and DDS were a script (which they are not), the DDS is not assigned to an element, but rather is applied to the document as a whole. (Birnbaum Dec. ¶ 40).

As described in the '789 patent, by assigning a script to a particular object, various benefits and advantages may be obtained, including, for example (see e.g., col. 5, lines 27-31 and col. 6, lines 38-51):

- automatically receiving data, e.g., data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.
- automatically sending or requesting data.
- reassigning data received to the graphic object associated with it.

- forwarding the graphic object associated with itself to a receiver together with the data requested, received and reassigned by itself.

The AS/400 machine described in the IBM reference cannot perform these functions. (Birnbaum Dec. ¶¶ 41, 42). Indeed, the architecture of the AS/400 described in the IBM reference would have to be entirely reconfigured and the code totally rewritten to perform these operations. (Birnbaum Dec. ¶ 43).

Because independent claims 1, 22, and 38 are allowable over the IBM reference, dependent claims 2-5, 17, 20, 21, 23-26, and 39-41, which depend directly or indirectly thereon, are likewise allowable.

II. Ground of Rejection No. 2: IBM in view of Interleaf

In the Office action, the Examiner rejected claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, and 48-49 under 35 U.S.C. § 103(a) as obvious over the IBM reference in view of Interleaf. See Request pp. 5, 7-8, and Exhibit H.

As explained in more detail below, it would not have been obvious in 2000 to combine the IBM and Interleaf references, and in any event, such combination would not have resulted in any of the claimed inventions.

A. There is no Prima Facie Case of Obviousness

A prima facie case of obviousness requires a showing that the invention as claimed would have been obvious at the time of the invention to one having ordinary skill in the art.

Although, the Examiner "can take account of the inferences and creative steps that a person of ordinary skill in the art would employ," *KSR International Co. v. Teleflex Inc.*, 127 S.Ct. 1727, 1741 (2007), it is critically important when undertaking this analysis to keep firmly in mind that "the person of ordinary skill in the art is an objective legal construct presumed to think along conventional lines without undertaking to innovate, whether by systematic research

or by extraordinary insights." Life Technologies, Inc. v. Clontech Laboratories, Inc., 224 F.3d 1320, 1325 (Fed. Cir. 2000); Std. Oil Co. v. Am. Cyanamid Co., 774 F.2d 448, 454 (Fed. Cir. 1985) ("A person of ordinary skill in the art is also presumed to be one who thinks along the line of conventional wisdom in the art and is not one who undertakes to innovate, whether by patient, and often expensive, systematic research or by extraordinary insights, it makes no difference which." (Emphasis added)). This person does not look for exceptionally creative solutions, and does not consider how to completely redesign existing software to arrive at novel, and admittedly extremely useful products that were previously unknown. In short, the person of ordinary skill in the art does not "think outside the box."

Indeed, as explained in MPEP (§ 2142):

> To reach a proper determination under 35 U.S.C. 103, the examiner must step backward in time and into the shoes worn by the hypothetical "person of ordinary skill in the art" when the invention was unknown and just before it was made. In view of all factual information, the examiner must then make a determination whether the claimed invention "as a whole" would have been obvious at that time to that person. Knowledge of applicant's disclosure must be put aside in reaching this determination, yet kept in mind in order to determine the "differences," conduct the search and evaluate the "subject matter as a whole" of the invention. The tendency to resort to "hindsight" based upon applicant's disclosure is often difficult to avoid due to the very nature of the examination process. However, impermissible hindsight must be avoided and the legal conclusion must be reached on the basis of the facts gleaned from the prior art.

Accord, *KSR* at1742 ("A factfinder should be aware, of course, of the distortion caused by hindsight bias and must be cautious of arguments reliant upon ex post reasoning. See *Graham* [*v. John Deere Co. of Kansas City*], 383 U.S. [1], 36 [1966] [] (warning against a 'temptation to read into the prior art the teachings of the invention in issue' and instructing courts to "'guard against slipping into the use of hindsight'"")); MPEP § 2141 II (Office personnel are factfinders).

The field of the invention of the '789 patent is, printing software generally, and in particular, interpretation or processing of print data streams at a printer or print server. (Birnbaum Dec. ¶ 56). A person of ordinary skill working in the relevant field in May 2000 (the

earliest priority date) or as late as May 2001 (the PCT filing date), would have been a person with a computer science degree (typically, a bachelor's degree), and approximately 2-4 years of experience. (Birnbaum Dec. ¶ 58). This experience may typically have been acquired working at a company making printers or printer software, such as Xerox, Hewlett-Packard, etc. (Birnbaum Dec. ¶ 59).

1. Defining the Field of the Invention and One of Ordinary Skill

As explained in MPEP § 2142, the Examiner must step backward in time and into the shoes worn by the hypothetical person of ordinary skill in the art and determine whether the claimed invention as a whole would have been obvious at that time to that person.

First, the field of the invention and the person of ordinary skill in the art must be defined. That determination is based on what is claimed. As the MPEP explains (§ 904.01(c)) (emphasis added):

> It depends upon the necessary essential function or utility of the subject matter covered by the claims, and not upon what it is called by the applicant.

All claims at issue expressly cover transformation of print data, in particular parsing input print data streams. The field of the invention is not as Samsung suggests, document creation (Interleaf) or user interfaces (Lieberman), because the structure and function of those references are totally unrelated to the subject matter and claims of the '789 patent, and therefore are plainly non-analogous arts. See MPEP § 2141.01(a) II.

Obviously, the relevant field cannot be the entire field of computer science, which essentially is what Samsung is arguing.

2. The IBM and Interleaf References are in Entirely Different Fields

These principles mean that a person of ordinary skill would not have combined the Interleaf references with the IBM reference.

The IBM reference describes how data from an application running on the AS/400 may be formatted (e.g., using a DDS) and sent to a printer. It describes providing print capability to networks controlled by a central server, like the AS/400. (Birnbaum Dec. ¶ 61).

However, the Interleaf references deal with "document creation" and manipulation software (Interleaf, p. 75).[4] It is completely unrelated to the field of the invention, i.e., printing software generally, and interpretation of print data streams at a printer or print server in particular. (Birnbaum Dec. ¶ 62). The Examiner points to Interleaf's brief discussion of printing an Interleaf document. (Office action, pp. 18, 27). However, this merely refers to a print function (present in any user application) to generate a print stream, presumably in a known print data stream, such as a page definition language. It is completely irrelevant to the Interleaf reference what happens to the print data downstream. That is, Interleaf does not involve interpretation of the print data streams at a printer or print server. (Birnbaum Dec. ¶ 63).

As a practical matter the Interleaf reference addresses issues that are unrelated to the '798 patent. For example, people involved in document creation software and those working in printer software or programming would typically have been working in different divisions or departments, or more likely in different companies altogether. (Birnbaum Dec. ¶ 60). One of ordinary skill in the field of printing would not have looked to the Interleaf reference, and its disclosure of document creation, in order to solve problems involving print data streams.

The USPTO itself ascribed the '789 patent and the Interleaf patent to entirely different fields: the U.S. Class/Subclass assigned to the '789 and Interleaf patents (and their fields of search) highlight their differences:

4 The Interleaf reference is discussed in detail in connection with Ground of Rejection No. 5.

'789 patent Classifications	**Description**
101/484	**PRINTING** **PROCESSES**
400/61, 62, 63; 400/76	TYPEWRITING MACHINES INCLUDING CONTROL OF FORMAT AND SELECTION OF TYPE-FACE BY PROGRAMMED CONTROL-SYSTEM INCLUDING CONTROL OF FORMAT BY PROGRAMMED-CONTROL-SYSTEM
358/1.1, 1.9, 1.15, 1.16, 1.17, 1.18	FACSIMILE AND STATIC PRESENTATION PROCESSING STATIC PRESENTATION PROCESSING (E.G., PROCESSING DATA FOR PRINTER, ETC.)

Interleaf Patent	**Description**
717/139	**DATA PROCESSING: SOFTWARE DEVELOPMENT, INSTALLATION, AND MANAGEMENT** **SOFTWARE PROGRAM DEVELOPMENT TOOL (E.G., INTEGRATED CASE TOOL OR STAND-ALONE DEVELOPMENT TOOL)**
715/234	DATA PROCESSING: PRESENTATION PROCESSING OF DOCUMENT, OPERATOR INTERFACE PROCESSING, AND SCREEN SAVER DISPLAY PROCESSING PRESENTATION PROCESSING OF DOCUMENT

See MPEP § 2141.01(a) II ("While Patent Office classification of references and the cross-references in the official search notes of the class definitions are some evidence of 'nonanalogy' or 'analogy' respectively, the court has found 'the similarities and differences in structure and function of the inventions to carry far greater weight.'"). Here there is no overlap between the classification – and, as explained above, no overlap between either the structure or the function of the prior art references and the '789 patent.

The Examiner, in adopting Samsung's proposed rejections, stated that the "prior art references are analogous as both address object-oriented publishing systems with scripting functionalities." (Office action, p. 11, and elsewhere).

Samsung has proposed, and the Examiner has incorrectly adopted, two different fields of the inventions depending on which prior art reference Samsung was trying to match up; Samsung's two fields of the invention are inconsistently defined as follows:

(1) "object oriented document publishing systems with scripting functionality." [5]

(2) "scripting interfaces for object systems"[6]

Obviously, the field of the invention is defined by the patented invention itself, not the prior art an infringer is seeking to combine, as was clearly the case here.

The fact that Samsung has proposed two different fields of the invention itself demonstrates that Samsung's conclusions are baseless.

Moreover, (1) is not a definition of a field of art that existed *prior* to the patent's priority date, but rather a hindsight rephrasing of keywords from the '789 patent, which does not reflect the realities of the programming world. See MPEP § 2142 (cited above). Number (2) is equally odd. The patented technology is not a generic "object system" – even assuming that phraseology has ever existed anywhere but Samsung's papers. CCP knows of no field that existed in 2000 described as set forth in (1) or (2). (Birnbaum Dec. ¶ 57).

It is worth pointing out that that the USPTO's classifications cited above for the Interleaf patent do not include either of these supposed fields, nor does the classification for the '789 patent.

As explained above, the '789 patent deals with specific problems and claims specific solutions to those problems. Samsung's "fields" are made up for the purpose of this reexamination and are improper.

---

[5] See Exhibit H (p. 11; to combine IBM and Interleaf, and adopted by the Examiner on pp. 11, 17 and 41).

[6] See Exhibit J (p. 21; to combine Interleaf and Lieberman, and adopted by the Examiner on pp. 26, 33 and 39).

In any event, as discussed above, the IBM reference does not disclose object-oriented programming, nor does it have scripting functionalities, and the Interleaf documents are not printing system. These references have been shoehorned into Samsung's artificial "fields" for the sole purpose of trying to prove invalidity in this reexamination.

In a related instance, then Chief Judge Markey of the Federal Circuit held:

> Appeals in patent cases should not be mere games played with pieces of paper called references and the patent in suit. Lawsuits arise out of the affairs of people, real people facing real problems. So here, the technical problem out of which grew the present litigation was real. It was solved by inventor Cardeiro.

Rosemount, Inc v. Beckman Instruments, Inc., 727 F.2d 1540, 1544 (Fed. Cir. 1984).

As in the *Rosemount* case, Samsung is indeed "play[ing] with pieces of paper called references" that do not relate to the problem that CCP solved. Samsung is looking for keywords in documents that do not perform the claimed functions, slapping these references together, and concluding that someone else would have invented the technology at issue earlier. The fact is no one did.

The real question is whether prior to CCP's patented and claimed inventions would it have been obvious to arrive at those inventions, not whether words on one page match words in a patent claim. The subsequent question is whether related technology would have been combined by a person of ordinary skill in a way to arrive at the claimed invention, not whether disparate technical disclosures stuck together by a defendant in a patent lawsuit have terminology similar to the asserted patent.

When understood in the proper context, the Interleaf reference (and patent) are in an entirely different field of technology than the '789 patent. One of ordinary skill – as properly defined – would not have referred to the Interleaf references or combined them with the IBM reference to arrive at the inventions of the '789 patent. (Birnbaum Dec. ¶ 64).

3. There Would Have Been no Motivation to Combine IBM and Interleaf

The Examiner, again adopting Samsung's request, purported to explain the motivation for combining IBM with Interleaf as follows:

> Combining the print process of IBM that parses and transforms a print stream into an object-oriented format, and the print-scripting process of Interleaf are combining well known printing techniques to yield predictable results.

Office action, pp. 11-12, citations omitted. As discussed below, this is pure hindsight reasoning, and is not based in technological realities at the time of the invention. Moreover, there is no "print-scripting process of Interleaf". (Birnbaum Dec. ¶ 66). Neither of the Interleaf references discloses that functionality. (Id.). This, therefore, is a flawed argument.

Conceivably, after extensive redesign, the Interleaf and IBM references might have been linked, but not combined. (Birnbaum Dec. ¶ 67). That is, although Samsung has not bothered to show it would even be possible, let us assume, *arguendo*, that the Interleaf system were operating on a PC connected to an AS/400 midrange computer.

A user *might* be able to operate the Interleaf system on the PC, and then send a document to the AS/400 to print. The AS/400 would then process the data, for example, by formatting it using a DDS. The result, however, would not result in the inventions claimed in the '789 patent because any object-oriented formatting, including any scripts, in the Interleaf "active document" would be lost prior to generating the print data stream that would be transmitted to the AS/400. Thus, even in a "linked" system, the AS/400 print data stream could not have either: (a) object-oriented formatting or (b) scripts disclosed in the Interleaf references. (Birnbaum Dec. ¶ 68).

Indeed, IBM's Figure 177 is instructive, showing how a PC client, e.g., operating Interleaf software, would have interacted with the AS/400.

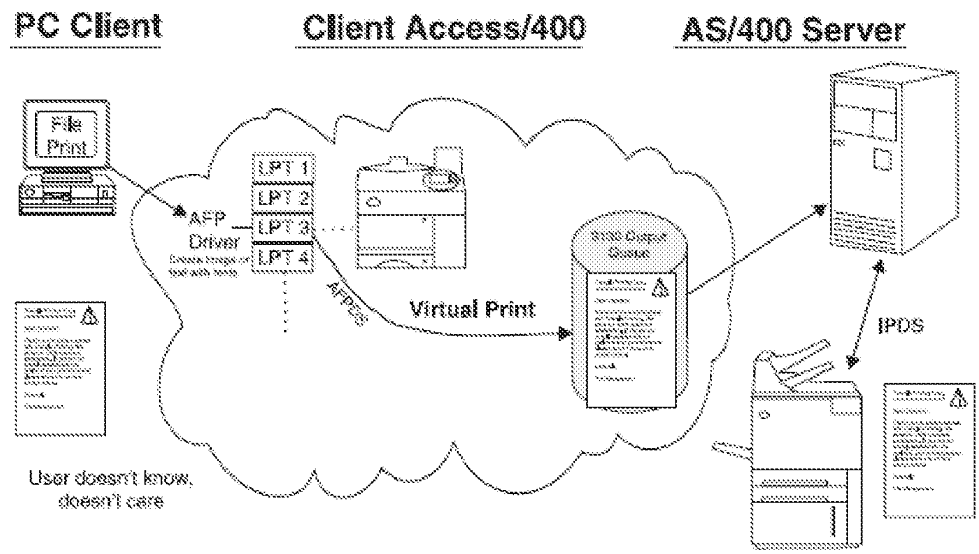


Figure 177. Printing PC applications on AS/400 IPDS printers

Samsung's first purported motivation is that "Interleaf suggests the combination because it is explicitly designed to leverage outside printing and publishing functionality," citing to Interleaf, p. 82, which refers to external publishing functionality, such as printing (Office action, p. 11). It is no accident that "presentation and editing functionality," i.e., the primary subjects of the Interleaf reference, are contrasted with <u>external</u> functionality, such as printing, which is peripheral. In any event, in a "combined" IBM+Interleaf system, Interleaf documents may be formatted and then sent to the AS/400. However, in order to combine the relevant features of Interleaf and IBM, one would have had to redesign the AS/400 AFP by taking the entire object-oriented structure, together with the scripts, described in Interleaf, and importing them onto the AS/400, where the printing facilities reside. (Birnbaum Dec. ¶ 69). There is absolutely no suggestion to do this – nor is there any reason to do this. It would result in an entirely reconstructed system. See, e.g., MPEP § 2143.01(VI) ("If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims *prima facie* obvious.")

Samsung's second purported motivation is that "IBM provides external printing and publishing functionality. . . IBM also teaches the desire to control outside printing and publishing functionality. . ." (Office action, pp. 11-12). The "external" functionality of Interleaf refers to printing, whereas the "external" functionality of IBM refers to staplers and binders, neither of which requires scripting functionality or object oriented code. (Birnbaum Dec. ¶ 70). Again, common words (without a common meaning) cannot, contrary to Samsung's assertions, be the basis for a motivation to combine the references.

Moreover, it is not enough to state that it would have been obvious to combine two references. As the Supreme Court held in *KSR*: "'Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness'"; see also. MPEP § 2142 ("rejections on obviousness cannot be sustained with mere conclusory statements.")

As discussed above, the AS/400 operates on data using a compiled DDS, not a script. It would not have been obvious – even in light of the Interleaf reference – to modify the AS/400 to use scripts. (Birnbaum Dec. ¶ 71). The AS/400 is a centralized platform, and therefore, would not have benefitted from the platform-independence of scripts. (Birnbaum Dec. ¶ 72). On the contrary, scripts may often be slower to execute than object code, because they must be interpreted in real time. (Birnbaum Dec. ¶ 73). Samsung has not shown that one or ordinary skill would have been motivated to replace the compiled DDS with a script to be interpreted at run time.

Samsung's conclusion is that a "person of ordinary skill in the art would be motivated to 'take full advantage' of the printing and publishing capabilities of the IBM system by combining it with the Interleaf system." (Office action, pp. 11-12). This is not a motivation; it is a play on words, devoid of technological substance, and assumes the conclusion. There are many, many ways to "take full advantage" of IBM's capabilities that have nothing to do with the inventive technology (e.g., using a faster printer, a color printer, larger or faster memory).

4. The IBM and Interleaf References Even if Combined Would Not have Resulted in the Inventions of the '789 Patent

Finally, even if the Interleaf system operating on a PC were linked to an AS/400 midrange computer, it would not have resulted in the present invention. A user may operate the Interleaf system on the PC, and then upon entering a "print" command in the application, send the output document to the AS/400 to print. (Birnbaum Dec. ¶ 75). However, the print output of such a PC would immediately have been converted to the structured AFP format in order to be manipulated by the AS/400 (Id.) The IBM reference states:

> Network applications are generating document data streams, such as Postscript, and image formats, such as GIF and TIFF. Data in these formats can be converted to AFP and then stored or printed from the AS/400. (IBM, p. 8, emphasis added).
>
> PostScript is a data stream generated by many PC and network station applications. . . If that PostScript data stream is being spooled to the AS/400, the data stream must be converted to AFP data stream in order to print on an IPDS printer. (IBM, p. 16, emphasis added).

However, as explained above, this conversion into the structured AFP data stream is not the purported "parsing" to which Samsung referred (IBM reference, p. 194). Nor would using the Interleaf software on a PC client connected to an IBM AS/400 cause the AS/400 to store graphically representable objects in object-oriented format, or assign a script to an object. Adding Interleaf to the "front end" of the IBM reference – simply results in printing documents as disclosed in the IBM reference. Nothing in the print stream is changed: there still are no objects, no scripts assigned to objects and no parser. (Birnbaum Dec. ¶ 76).

Regarding the dependent claims, as discussed, the IBM reference and the Interleaf reference do not individually disclose the elements of claims 2-7, 11, 12, 17, 20-28, 32, 33, 38-44, 48, and 49, and therefore, the combination of IBM and Interleaf does not disclose these claims.

B. Objective Evidence of Non-Obviousness Overcomes Any Purported Showing to the Contrary

Even if there were a prima facie case of obviousness, it is far outweighed by objective evidence of non-obviousness presented in the Widuch and declarations.

Secondary considerations of non-obviousness must be considered. *Transocean Offshore Deepwater Drilling, Inc. v. Maersk Contractors USA, Inc.*, 617 F.3d 1296, 1305 (Fed. Cir. 2010) ("[T]he district court must *always* consider any objective evidence of nonobviousness presented in a case.") (Emphasis in original); see MPEP § 716.01(a) ("Affidavits or declarations. . . containing evidence of criticality or unexpected results, commercial success, long-felt but unsolved needs, failure of others, skepticism of experts, etc., must be considered by the examiner in determining the issue of obviousness of claims for patentability. . .") (Emphasis added); § 2154 ("Evidence pertaining to secondary considerations must be taken into account whenever present[.]") (Emphasis added).

> Evidence of secondary considerations may often be the most probative and cogent evidence in the record. It may often establish that an invention appearing to have been obvious in light of the prior art was not. It is to be considered as part of all the evidence, not just when the decisionmaker remains in doubt after reviewing the art.

Stratoflex, Inc. v. Aeroquip Corp., 713 F.2d 1530, 1538 (Fed. Cir. 1983) (emphasis added, internal quotation marks omitted).

"Once the applicant has presented rebuttal evidence, Office personnel should reconsider any initial obviousness determination in view of the entire record." MPEP § 2141(V). This is the stage where we are now.

The current obviousness determination cannot survive where the owner of the allegedly invalidating prior art spent millions of dollars on the patented technology and where the requestor also paid millions of dollars in licensing fees and praised the patented product. *WMS Gaming, Inc. v. Int'l Gaming Tech., Inc.*, 184 F.3d 1339, 1359 (Fed. Cir. 1999); *see also, Ruiz v. A.B. Chance Co.*, 234 F.3d 654 (Fed. Cir. 2000) (remanding where the district court failed to consider objective considerations).

Because this objective evidence arises out of economic and business realities, and is not subject to the vagaries of litigation inspired technical arguments, it often is the most probative and cogent evidence in the record.

Indeed, the objective evidence of non-obviousness may overcome a prima facie case of obviousness. *Gambro Lundia AB v. Baxter Healthcare Corp.*, 110 F.3d 1573, 1580 (Fed. Cir. 1997) (reversing finding of invalidity where district court failed to consider objective evidence of non-obviousness, including commercial success and failure of others); *Ruiz* at 667; *Hughes Tool Co. v. Dresser Industries, Inc.*, 816 F.2d 1549 (Fed. Cir. 1987) (affirming validity based on objective evidence of non-obviousness); *Simmons Fastener Corp. v. Ill. Tool Works, Inc.*, 739 F.2d 1573, 1575, (Fed. Cir. 1984) (reversing holding of invalidity and finding that the objective evidence overcame the lower court's decision on obviousness, holding that "[o]nly after all evidence of nonobviousness has been considered can a conclusion on obviousness be reached.").

Indeed, the Federal Circuit has reversed obviousness rejections for failure to sufficiently consider objective evidence:

> when an applicant puts forth relevant rebuttal evidence, as it did here, the Board must consider such evidence. The claimed composition cannot be held to have been obvious if competent evidence rebuts the prima facie case of obviousness. <u>By failing to consider the submitted evidence, the Board thus committed error</u>.

In re Sullivan, 498 F.3d 1345, 1353 (Fed. Cir. 2007) (emphasis added). Likewise, in another case:

> appellants submitted extensive evidence of peer recognition, long-felt need, and commercial interest. Yet the Board's treatment of the rebuttal documents impels us to the conclusion that the Board did exactly that which *Rinehart* warns against: they viewed each piece of rebuttal evidence solely "on its knockdown ability". Under the Board's approach the *prima facie* case took on a life of its own, such that each fact presented in rebuttal, when it was evaluated at all, was evaluated against the conclusion itself rather than against the facts on which the conclusion was based. The *prima facie* case remained 'set in concrete'.
>
> <u>This procedure, and the conclusion of obviousness flowing from this procedure, are thus flawed</u>.

In re Piasecki, 745 F.2d 1468 (Fed. Cir. 1984) (emphasis added).

Samsung's use of the patented technology certainly demonstrates commercial success; Samsung's and IBM's comments show praise in the industry; and Samsung's illegal use following termination of the IBM License and its appropriation of the name "JScribe" also shows copying, all important objective of non-obviousness that easily overcome any *prima facie* claim of obviousness. This evidence must be considered in any obviousness analysis, *Transocean* at 1305 (Fed. Cir. 2010); see MPEP §§ 716.01(a), 2154, and changes the result here.

1. Commercial Success and Licensing Supports Validity of the '789 Patent

As discussed above, in 2004 IBM's German subsidiary entered into the IBM License for CCP's JScribe® technology, covered by the '789 patent, and committed IBM to pay tens of millions of dollars. (Widuch Dec. ¶ 7).

Konica-Minolta also paid IBM millions of dollars as a lump sum for the technology and agreed to pay a running royalty to IBM, of which CCP received 50%. (Widuch Dec. ¶ 8).

Samsung paid IBM several million as a lump sum and was supposed to pay IBM a running royalty for each device that included the patented JScribe® technology. (Widuch Dec. ¶ 10).

2. Praise in the Industry Supports Validity of the '789 Patent

In *Gambro Lundia AB v. Baxter Healthcare Corp.*, 110 F.3d 1573, 1579 (Fed. Cir. 1997), the infringer (Baxter),

> touted the advantages of [the patented feature] AUTO-ADJUST, as it termed automatic recalibration during dialysis, in the advertising for the allegedly infringing Baxter SPS 550 machines. Baxter's recognition of the importance of this advance is relevant to a determination of nonobviousness.

Accord Eli Lilly & Co. v. Zenith Goldline Pharmaceuticals, Inc., 471 F.3d 1369 (Fed. Cir. 2006) (pointing to the record showing a number of awards as indicators of industry acclaim). See also, *In re Tiffin*, 443 F.2d 394, 400 (CCPA 1971):

> Recognition by the trade is the best and most persuasive evidence that can be offered. The tribute of those engaged in the industries affected, especially when the tribute is evidenced by the payment of substantial royalities [sic], is by far the most persuasive and unimpeachable evidence that can be offered to support the asserted validity of patent claims in litigation. Self interest of the licensee can be relied upon to prevent the payment of royalties, except for discoveries of recognized merit, and the larger and more numerous the royalty payments, the stronger the inference that payment is made only after the licensee has satisfied itself that the invention is meritorious and the claims covering the process or product are valid.

Upon granting Samsung a sub-license to the JScribe® technology, IBM and Samsung issued a joint press release that stated: "'JScribe', the printing optimized middleware of IBM will be launched in the Samsung digital printing devices such as printers and multifunction products." (Emphasis added) (Widuch Dec. Ex. B). It is unclear which of the two companies (or both) was responsible for drafting this document, but JScribe® has always been a CCP product. Nevertheless, IBM and/or Samsung took credit for the innovative technology.

In 2006, IBM awarded CCP its "Best Seller Award," a recognition of CCP's groundbreaking work on JScribe technology. (Widuch Dec. Ex. C).

In June 2007, CCP's JScribe® product won the European Commission China Information and Communications Technologies Innovator Award (ChinICT). (Widuch Dec. Ex. D).

In about 2008, Samsung issued another press release entitled: "New Samsung MFP with Advanced Features Offers Easy Integration for Efficient, Simple and Reliable Printing" that discussed the many advantages of the CCP product that Samsung characterized as "Samsung's JScribe™". Samsung obviously does not own the trademark to JScribe. But, it is interesting that Samsung appropriated this important technology as its own, technology that it now belittles. (Widuch Dec. Ex. E, p. 2).

In that press release, Samsung also proclaimed the benefits of CCP's JScribe® product (Widuch Dec. Ex. E, p. 2):

- "open platform technology means these new features can be added quickly and easily in response to the individual needs of the customer."
- "Corporate users can maximize their printing investment by customizing the device to their specific requirements."

In about 2008-09, Samsung gave the patented JScribe® technology effusive praise in a white paper (Widuch Dec. Ex. F):

- "highly flexible and portable to almost any microprocessor-based system" (p. 1);
- "A simple control script can serve as the device driver and can control a broad range of devices including: fingerprint readers, Card readers, RFID printers, RFID readers, label printers, Slave printers." (p. 2).
- "Print job filter objects allow JScribe applications to recognize incoming print data and, if required, modify the print sequences before the print data reaches the MFP's RIP [Raster Image Processor]." (p. 2).
- "In the past, modifying the printer code was either impossible or extremely expensive. This left the user with very few choices when a new printer-related business task emerged." (p. 3).
- "Advantages over Other Embedded Platforms Other application development platforms for MFPs use an interpreter to translate the software into a language understood by the hardware. JScribe uses the MFP's native code to perform critical functionality (such as managing print or image data). This allows JScribe applications to parse print data directly without being limited by high-level data manipulation." (p. 3).

- "Compared to the traditional firmware development process, creating a JScribe application requires much less programming knowledge and can be accomplished in a fraction of time." (p. 4).
- "JScribe allows Samsung to provide customers with custom solutions that meet their enterprise requirements. In the past, fulfilling these customer requirements was restricted by cost and time intensive firmware modifications requiring R&D resources. JScribe allows us to significantly reduce these costs and the R&D resources needed to successfully develop prototypes and pilot applications." (p. 5).

Samsung identified seven lines of printers that include CCP's JScribe® product and a number of customers that used thousands of Samsung printers with JScribe® technology installed, including "a large government institution in the US". (pp. 4-6). In CCP's view, all the deals cited in the Samsung White Paper (Widuch Dec. Ex. F) could only have been done because of the JScribe® technology in Samsung's devices. (Widuch Dec. ¶ 17).

3. Copying by Others Supports Validity of the '789 Patent

In 2009, CCP terminated the License with IBM, and with it any rights that Samsung may have had under that agreement. (Widuch Dec. ¶ 19).

Now, even after having been sued, and after bringing on this reexamination, Samsung's web site continues to promote the advantages of "JScribe technology": "As printers have gone from only printing to complicated multi-function machines, Samsung has provided JScribe software to support all the new functions and features." (Widuch Ex. G).

* * *

To summarize: at least two critical and undeniable facts stand out. First, IBM, on whose technology Samsung relies in this reexamination, spent million dollars to acquire rights to the patented technology and paid on-going royalties to CCP for the privilege of licensing and sub-

licensing the patented technology. Second, Samsung, the requester here, paid IBM, and IBM paid the patent owner, CCP, millions of dollars for the rights to license and use the patented technology. (Widuch Dec. ¶¶ 7, 10).

Despite all this contrary evidence, Samsung now says that IBM's AS/400 software is identical to the '789 patent (i.e., IBM anticipated the claimed subject matter) and rendered obvious the innovations claimed in the '789 patent. But the truth of the matter is that in 2004, four years *after* the patent's priority date, and five or more years after publication of the IBM reference, IBM paid handsomely to license CCP's software covered by the patent at issue. (Widuch Dec. ¶ 7).

As a matter of common sense, if the innovations of the '789 patent were so obvious (as Samsung now claims), and IBM already had identical or similar software, neither Samsung nor IBM would have paid anything to license CCP's software.

III. Ground of Rejection No. 3: IBM in view of Interleaf and Interleaf Patent

The Examiner rejected claims 1-14, 16-18, 20-35, 37-51 and 53 under 35 U.S.C. § 103(a) as obvious over the IBM reference in view of Interleaf and the Interleaf Patent. See Request pp. 10-12, and Exhibit I.

The Interleaf patent does not add relevant subject matter to the Interleaf reference. The Interleaf patent, too, deals with document creation and editing. (Birnbaum Dec. ¶ 84). At most, the software described in the Interleaf patent may generate a print stream, but it does not interpret or transform a print stream, as claimed in the '789 patent. Therefore, the same reasons, presented above, why the references would not have been combined, and in any event, would not have resulted in the inventions of the '789 patent if combined, are applicable here. The objective evidence of non-obviousness is likewise applicable.

IV. Ground of Rejection No. 4: IBM in view of Interleaf/Interleaf Patent and Lieberman

The Examiner rejected claims 15, 19, 36, and 52 under 35 U.S.C. § 103(a) as obvious over the IBM reference in view of Interleaf and further in view of the Interleaf Patent and Lieberman. See Request pp. 10-13, and Exhibit I. Patentee traverses the rejection for at least the following reasons.

Lieberman is about automating user interaction by using scripts. The paper reports "experiments in developing agent software that works with existing unmodified commercial applications and agents that work across multiple applications."

Neither Samsung's request for reexamination nor the Office action explains how Lieberman relates to the field of the invention, interpretation of print data streams at a printer or print server. Lieberman has nothing to do with printing, print streams, printers, or print servers. (Birnbaum Dec. ¶ 100). There would have been no reason for one of ordinary skill in the art to refer to the Lieberman reference, or to use its teachings to solve the problems addressed in the '789 patent.

Accordingly, the obviousness determination based on any combination involving the Lieberman reference is simply wrong – and is the result of improper hindsight analysis.

V. Ground of Rejection No. 5: Interleaf

The Examiner adopted the Requester's proposed rejection of claims 1-2, 4-7, 11-12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 under 35 U.S.C. § 102(b) as anticipated by Interleaf. See Request pp. 10-14, and Exhibit I.

Generally, the objective of the Interleaf system is to build an enhanced user interface (UI). ("Use of this natural UI makes this produce easy to learn and hence extremely popular." Interleaf, p. 76). Interleaf is a "document creation component" (Interleaf, p. 75). As explained above, the '789 patent addresses an entirely different part of the workflow, namely, the printing

stage. Consequently, there are at least four fundamental differences between Interleaf and the '789 patent. Interleaf does not:

- disclose a method for the transformation of print data streams;
- read in an input print stream;
- parse an input print data stream; or
- produce an output print data stream based on a transformed input data stream.

(Birnbaum Dec. ¶ 103).

A. Interleaf does not disclose transformation of print data streams

Claims 1, 22, and 38 recite a method, computer-readable medium, and computer signal for the transformation of digital print data streams. The Examiner purported to find this claim element at Interleaf, p. 76: "I6 also has font and other style tokens that can be placed in the middle of a component text stream to change style without requiring an inline component." In addition, the Examiner has further pointed to pp. 81-84, regarding changing objects in a document, e.g., changing color or size or position of the object. But these refer to editing a working document in the user interface, not printing it; that is, these changes are performed to edit or create the document, not to print it. (Birnbaum Dec. ¶ 104).

These portions of Interleaf do not describe the transformation of one print data stream (input) into another data stream (output). Rather, they merely describe users creating or editing a document, not what happens at print time. Therefore, they cannot describe transformation of a print data stream. (Birnbaum Dec. ¶ 105).

B. Interleaf does not read in an input print stream

The claims recite that an input print data stream is read in. The Examiner purported to find this element at Interleaf p. 84 ("If a document contains active objects within the document file stream. . . these become activated when the document is opened programmatically or by the

user for viewing, editing, or printing." Emphasis added.) This quote therefore refers to a document file stream, not a print data stream. The document file stream is merely what is read in by the Interleaf system when the Interleaf document is opened. It is not a print data stream, as recited in the claims. (Birnbaum Dec. ¶ 106).

Opening an Interleaf document is akin to opening a Word document. However, this operation does not create a print data stream. Print data are only generated when the print function is activated, e.g., a print icon in Word is clicked. (Birnbaum Dec. ¶ 107).

In order to print a document, the Interleaf reference may conceivably generate a print data stream as its output, but it does not read in a print data stream, as claimed in the '789 patent. This fundamental difference alone distinguishes Interleaf from the '789 patent. (Birnbaum Dec. ¶ 108).

C. Interleaf does not parse an input print data stream

The claims recite that the input print data stream are analyzed by a parser for graphically representable objects. The Examiner purported to find this element in the Lisp interpreter operating on the Interleaf document. As stated in the portions quoted by the Examiner regarding an "active load hook" (pp. 81-82, 85), the Lisp interpreter operates on a document file stream, or, more precisely, on the Lisp objects in the document. However, as discussed above, the document file stream is not a print data stream. Therefore, even assuming the Lisp interpreter were a parser, it does not parse a print data stream. (Birnbaum Dec. ¶ 109). In addition, the Lisp interpreter (by definition) only interprets LISP scripts, and nothing else. Therefore, it is incapable of interpreting print data streams, as contrasted from the parser of the claims of the '789 patent, which analyzes an input print data stream. (Birnbaum Dec. ¶ 110).

D. Interleaf does not produce an output print data stream based on a transformed input print data stream

The claims recite that "(iv) the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer, and (v) the objects thus transformed are combined into an output print data stream and are output." Emphasis added.

The Examiner pointed to "transformation capabilities used to control publishing," such as automatic pagination. Although Interleaf, like many other document creation tools, may eventually create a print data stream (most likely by using an appropriate printer driver) – Interleaf does not read that stream into and analyze it using a parser and split it up into objects. See e.g., claim 1 (preamble, (i) and (ii). Not having this capability, Interleaf obviously cannot transform objects and combine those transformed objects into an output print data stream. See e.g., Claim 1 (iv) and (v). (Birnbaum Dec. ¶ 111).

The Examiner stated "[o]pening the document for printing shows the system actually outputting the output print data stream." (Office action, p. 29). Simply, "opening" a document for printing, plainly does not: (1) read the print data stream; (2) parse it; (3) split it up into objects; (4) transform objects to control a printer; or (5) combine those transformed objects into an output data stream. (Birnbaum Dec. ¶ 112).

VI. Ground of Rejection No. 6: Interleaf in view of Lieberman

The Examiner rejected claims 15, 36, and 52 under 35 U.S.C. § 103(a) as obvious over the Interleaf reference in view of Lieberman. See Request pp. 13-15, and Exhibit J. Patentee traverses the rejection for at least the following reasons.

As discussed above, with reference to Ground of Rejection No. 4, Lieberman does not involve printing whatsoever. A person of ordinary skill in the relevant art would have had no reason in 2000 to combine IBM or the Interleaf references with Lieberman for any reason. (Birnbaum Dec. ¶ 120). Suggesting that this combination would have been made is a classic

exercise of hindsight analysis, using the patent at issue as template into which to place arbitrary prior work.

The secondary indicia of non-obviousness also show why this conclusion is unsupported.

Moreover, even if Lieberman and Interleaf were combined, for reasons explained above, the combination would not result in the claimed inventions (e.g., no parsing of an input print data stream). (Birnbaum Dec. ¶ 121).

VII. Grounds of Rejection No. 7: Interleaf in view of Interleaf Patent

The Examiner rejected claims 1-2, 4-14, 16-18, 22-23, 25-35, 37-39, 41-51, and 53 under 35 U.S.C. § 103(a) as obvious over the Interleaf reference in view of the Interleaf Patent. See Request p. 15, and Exhibit K. Patentee traverses the rejection for at least the following reasons.

The Interleaf Patent does not rectify the deficiencies of the Interleaf reference disclosed above with respect to Ground of Rejection No. 5, insofar as it, too, does not relate to transforming a print data stream. To purportedly find a script executed in the cases defined in the script, the Examiner referred to the following in the Interleaf Patent: "As explained above, any data object within the hierarchy may be associated with a method which, upon the occurrence of a certain event, will be implemented." (Col. 3, lines 37-39). However, this must be read in context. The entire paragraph reads:

> As explained above, any data object within the hierarchy may be associated with a method which, upon the occurrence of a certain event, will be implemented. For example, referring to FIG. 5, memorandum 32 is associated with an open method 64. When a user, via user interface 30, indicates that he desires to view memorandum 32, the open method 64 is implemented to open the memorandum file 31 and prepare image data representative of how the document would appear if printed. The image data is supplied to display device 24 which generates an image of the document as shown in FIG. 3. Similarly, when the user indicates the desire to close memorandum 32, close method 66 is implemented to store the electronic representation of the memorandum and remove its image from the display.

The occurrences upon which methods are implemented are in the context of user interactions with the document, e.g., user opening a document, clicking on an image, etc. They are not in a print data stream, nor are they implemented in the context of printing a document, as claimed in the '789 patent. (Birnbaum Dec. ¶ 123).

VIII. Ground of Rejection No. 8: Interleaf in view the Interleaf Patent and Lieberman

The Examiner adopted the proposed grounds of rejection of claims 15, 19, 36, and 52 under 35 U.S.C. § 103(a) as obvious over the Interleaf reference in view of the Interleaf Patent and further in view of Lieberman. Office action, p. 38. Patentee traverses the rejection for at least the following reasons.

As discussed above, with reference to Ground of Rejection No. 4, Lieberman does not involve printing whatsoever. A person of ordinary skill in the relevant art would have had no reason in 2000 to combine IBM or the Interleaf references with Lieberman for any reason. Suggesting that this combination would have been made is a classic exercise of hindsight analysis, using the patent at issue as template into which to place arbitrary prior work. (Birnbaum Dec. ¶ 135).

IX. Ground of Rejection No. 9: Interleaf Patent in view of IBM

The Examiner adopted the proposed grounds of rejection of claims 1-53 under 35 U.S.C. § 103(a) as obvious over the Interleaf Patent in view of IBM. Office action, p. 40. Patentee traverses the rejection for at least the following reasons.

The above remarks pertaining to the impermissible combination of the Interleaf reference and IBM are fully applicable here, and incorporated herein.

In addition to the above, the Examiner stated that "it is inherent that when printing, the objects of the active document to be printed are combined into an output stream." (Office action, p. 40). However, there is no overlap between the Interleaf Patent and the IBM reference. That

is, the print data stream produced by the Interleaf Patent as an output is handed off to the IBM reference as an input. (Birnbaum Dec. ¶ 137). It makes no sense to "combine" the references in the manner suggested by Samsung, or to apply processes of the IBM reference (print formatting and printer functionality) to those of the Interleaf Patent (document creation and editing).

Furthermore, reference in the Office action to various types of print data streams is irrelevant, because these are simply converted to AFP data streams for processing. (Birnbaum Dec. ¶ 138).

BRIEF DISCUSSION OF DEPENDENT CLAIMS

Claims 2, 23, and 39 add the limitation that objects can be combined to form super-objects. Regarding the IBM reference, nothing disclosed therein suggests that the data structures described (e.g., an image of an airplane) are linked to other objects with the properties listed above (including class hierarchy, assignment of methods to objects, and inheritance). (Birnbaum Dec. ¶ 47). As a matter of common sense if the objects to do not possess these properties, they cannot be linked together to form super-objects with those properties. (Birnbaum Dec. ¶ 48). Samsung's contention that "'Documents' and 'pages' are examples of super-objects of higher complexity' (Exhibit G, p. 14) falls flat because neither the document nor a page is an object, as claimed; not being objects, they obviously cannot be combined into a super-object. (Birnbaum Dec. ¶ 49). Similarly, Samsung's example that graphics, made up of lines, arcs, etc. and bar codes are super-objects fails for similar reasons. Lines, arcs, etc., are not objects, for reasons explained above, and combining them into a geometric shape does not create a super-objects, within the meaning of the '789 patent claims. (Birnbaum Dec. ¶ 50).

Regarding Interleaf, the Examiner pointed to component objects being changed, for example, "while the document is opening" (Office action, p. 12). However, as explained above, this bears no relation to transforming a print data stream, as claimed in the '789 patent. (Birnbaum Dec. ¶ 77). Similarly, the disclosure in the reference of a template document subclass with an 'open' method used for auto-localizing documents bears no relation to transforming a print data stream. (Birnbaum Dec. ¶ 113).

Regarding the Interleaf patent, Samsung points to col. 2, line 61 to col. 3, line 13, stating that the patent discloses a nested hierarchy of data objects. (Office action, p. 19). However, as with other portions of the Interleaf patent, this bears no relation to transforming a print data stream. (Birnbaum Dec. ¶ 85).

Claims 3, 24, and 40 recite that feedback messages that indicate that the output device, e.g., the printer has recognized a transformed graphic object in the output print data stream which cannot be output by the printer. Regarding the IBM reference (pp. 16-17 and 26), the feedback mechanism provided by IPDS does not have the functionality recited in these claims. IPDS allows the system to recognize whether the printer already has some of the elements (resources) needed for printing the document, or whether the printer was able to print the page or not. (Birnbaum Dec. ¶ 52). However, the IBM reference does not disclose analysis of feedback messages for error messages which indicate that the output device has recognized a transformed graphic object in the output print data stream which cannot be output by the printer, and that based on such a feedback message, splitting up the graphic object into objects of lower complexity that can be printed. (Birnbaum Dec. ¶ 53).

Claims 4, 25, and 41 recite that a graphically representable object is assigned at least one script which controls external devices. The IBM reference does not disclose scripts, scripts assigned to objects, or objects stored in object-oriented format. (Birnbaum Dec. ¶ 54). Therefore, the IBM reference as a whole cannot disclose that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which controls external devices, as recited. (Id.)

Regarding Interleaf, the Office action cites: (1) that active documents are defined as "structure documents. . . in which the objects. . . can be acted upon by, and can themselves act upon, other objects in the document or the outside world." (Interleaf, p. 78), and (2) that clicking on a face plays an audio name, etc. (Interleaf, p. 79). However, these portions are irrelevant to claims 4, 25, and 41, which have to do with use of objects in connection with transformation of a print data stream. (Birnbaum Dec. ¶ 78).

With regard to the Interleaf Patent, the Office action cites claim 9. A particular example is provided by a "phone directory that can call the person who's [sic] name you've selected; if the line is busy, it puts you into electronic mail." None of this relates to printing (there is no suggestion for the printer to call a person), and the AS/400's DDS would not have been capable of providing this functionality. (Birnbaum Dec. ¶ 86).

Claims 5, 26 and 42 recite that a graphically representable object is assigned at least one script which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

The IBM reference does not disclose scripts, scripts assigned to objects, or objects stored in object-oriented format. (Birnbaum Dec. ¶ 55). In addition, none of the cited portions of the IBM reference discuss automatically receiving data organized in an object-oriented manner, for example, from the Internet, data from XML documents or e-mails. (Id.)

Regarding Interleaf reference's use of LISP, nothing in IBM or Interleaf suggests such use in connection with an input print data stream generally, as recited in claims 5 and 26, or automatically receiving data, e.g., from web pages from the Internet, data from XML documents or else e-mails. (Birnbaum Dec. ¶ 79).

In the Interleaf patent, one purported application involves retrieving stock information and displaying it in text and charts; however, nothing suggests extracting the data at print time, that is, based on an input print data stream, much less doing so using the AS/400's DDS. (Birnbaum Dec. ¶ 87).

Moreover, any such capabilities in the Interleaf reference and patent are discussed with respect to the limited scope of document creation and editing. Once the document is sent to the printer, any such functionality is lost. That is, nothing in the references suggests obtaining such information during the printing process, i.e., after receiving an input print data stream. (Birnbaum Dec. ¶ 126).

Claims 6, 27, and 43 recite that the script automatically receiving data also requests this data automatically. The cited portions of the Interleaf reference and patent do not relate to use in

connection with an input print data stream. For example, the Examiner's reference to Interleaf's documents "not appear[ing] active to all users" further reinforces the point that the Interleaf reference relates to user editing and handling of documents, not their printing. (Birnbaum Dec. ¶ 80). Moreover, nothing suggests using the AS/400's DDS, as Samsung contends, to provide the functionality described in the Interleaf Patent, or that it is even possible to do so. (Birnbaum Dec. ¶ 88).

Claims 7, 28, and 44 recite that the script in turn reassigns the data received by it to the graphic object associated with itself, and prints out the graphic object assigned to itself together with the data requested, received and reassigned by itself.

The IBM reference does not disclose scripts, but merely the use of a compiled DDS on data. Accordingly, the functionality of the scripts recited in these claims cannot be performed with compiled DDS. Therefore, even if the Interleaf reference discloses such functionality, it would not have been possible to incorporate this into the AS/400, as suggested by Samsung. (Birnbaum Dec. ¶ 81).

The Office action points to the Interleaf Patent, which discloses a "customer receipt that automatically runs a 'frame grabber' and incorporates a photographic image of the purchaser." Again, this frame grabber in its incorporation into the receipt do not transpire based on an input print data stream, but during creation of a document. Nor is there any indication that the DDS used in the AS/400 would have been capable of performing such functionality. (Birnbaum Dec. ¶ 89).

Claims 8, 29, and 45 recite a one graphically representable object is assigned at least one script which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails." The Office action cites the Interleaf Patent, including an "urgent memo that integrates with voice annotation software so when it mails itself to someone, it announces its presence audibly" and a "document created from a database which updates the database if you make any changes in the imported document." These examples have no application in a print application. There is no reason to think that one of ordinary skill would have combined the

references to produce an urgent memo that when printed integrates with voice annotation software so when it prints itself, it announces its presence audibly, or that this would have been possible using the tools available from the IBM reference. (Birnbaum Dec. ¶¶ 90, 128).

Claims 9, 30, and 46 recite that "the script sends the graphic object associated with itself to [a] receiver." Regarding the Interleaf Patent, a "WYSIWYG document that can send itself over an electronic mail system" and "memos sent to you as a reminder of dates / events" have no application to a transformation of a print data stream, as recited in the claims. (Birnbaum Dec. ¶¶ 91, 129). Moreover, combining Interleaf with IBM would not maintain the email functionality, for the reasons described above, and DDS is incompatible with that ability.

Claims 10, 31, and 47 recite that "the script in turn reassigns the data received by it to the graphic object associated with it, and prints out the graphic object assigned to itself together with the data requested, received and reassigned by itself." The Office action cited the Interleaf Patent's example of a "stock-market information center that retrieves stock information and displays it in text and charts that can be automatically distributed or incorporated in other documents." See discussion of claims 5, 26, and 42, above. (Birnbaum Dec. ¶ 92, 130).

Claims 11, 32, and 48 recite that "at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which is executed in the case of the output of the object defined in the script." The IBM reference does not disclose scripts, but merely the use of a compiled DDS on data. Accordingly, the functionality of the scripts recited in these claims cannot be performed with compiled DDS. (Birnbaum Dec. ¶ 82). The Office action cited the Interleaf Patent at col. 6, lines 6-16, and col. 3, lines 37-39, but as discussed above, there is nothing to suggest combining this functionality with the IBM reference, which does not disclose scripts, or using this functionality to transform a print data stream. (Birnbaum Dec. ¶¶ 93, 131).

Claims 12, 33, and 49 recite that a graphically representable object is assigned at least one script, at least one case relating to the execution of the script being defined in the respective script, and occurring automatically, preferably without further influence from outside. The Examiner has pointed to a number of features of Interleaf reference and patent, but these operate

while the document is being edited or opened; they do not relate to manipulation of a print stream produced by an Interleaf document. In any event, Samsung has not shown that it would have been obvious or even possible to perform this functionality in the DDS used by the AS/400 system. (Birnbaum Dec. ¶¶ 83, 94, 119).

Claims 13, 34, and 50 recite that "that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time." **Claims 14, 35, and 51** recite that "the timer operates cyclically, that is to say it starts itself again upon expiry." The Office action provides a number of examples in Interleaf, e.g., a document that includes "Draft" before an announcement date, a reminder that appears on a screen based on a document due date, a weekly report built on a manager's desktop. First, these examples relate to the generation of a document, not an operation performed on a print stream based on a document. Second, simply comparing an announcement date to a current system date is not a timer function. It is not at all clear it would have been possible to implement either the date comparison or the timer functions on the AS/400. (Birnbaum Dec. ¶¶ 95, 132).

Claims 16, 37, and 53 recite that "the graphically representable objects stored in the memory in an object-oriented format, preferably also script objects, preferably before they are output in the output print data stream, are kept ready by an application interface to be read out, to be changed, to be deleted or to have new objects appended." The Examiner referred to various editing functions disclosed in the Interleaf Patent, and stated that these are purportedly performed "before they [the objects] are output in the output print data stream" because they happen before the document is printed. This interpretation conveniently ignores that fact that the claims expressly recite that the operations are performed <u>after</u> receiving an input print data stream. Nowhere does the Interleaf Patent disclose or render obvious these functions when performed after receiving an input print data stream, as required by the claims. (Birnbaum Dec. ¶¶ 96, 133).

Claim 18 recites a system that also has an operating station with display means and input means, which makes it possible for the graphically representable objects. . . preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream. The Interleaf Patent

does not disclose performing any such functions after receiving a print data stream. (Birnbaum Dec. ¶¶ 97, 134).

Claim 20 recites a "printer, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17." The Office action stated that "IBM teaches a printer . . . characterized in that it has a system for the transformation of digital print data streams." (Office action, p. 15). The AS/400 is not a printer, and a printer connected to the AS/400 would certainly not have a system for the transformation of digital print data streams – with or without the additional of any of the prior art references. Nor would it have been obvious based on either the IBM or Interleaf reference to take the functionality of the AS/400 main frame and import it into a printer. Doing so would be contrary to the centralized organization of the AS/400 system, and mostly like would have been technologically impossible to have printer's microprocessor perform the operations that had been carried out by a main frame computer. (Birnbaum Dec. ¶¶ 44-46).

The inventions of the '789 patent are new and would not have been obvious. Indeed, as shown by ample objective evidence, large sophisticated companies such as IBM and Samsung paid millions to license the patented technology. Samsung has sold hundreds of thousands of printers with the patented software embedded, and both IBM and Samsung gave the patented software effusive praise. All originally issued claims and the newly submitted claims should be confirmed. Please charge any fees associated with this paper to deposit account No. 50-3355.

Respectfully submitted,

/Guy Yonay/
Guy Yonay
Attorney/Agent for Applicant(s)
Registration No. 52,388

Dated: January 10, 2011

Pearl Cohen Zedek Latzer, LLP
1500 Broadway, 12th Floor
New York, New York 10036
Tel: (646) 878-0800
Fax: (646) 878-0801

Patent No. 6,684,789 Reexamination filed: July 16, 2010
Control No. 95/001,398

CERTIFICATE OF SERVICE

I hereby certify that on this 10th day of January, 2011, a true and correct copy of the following documents and their respective exhibits:

Amendment and Response to Office action

Declaration of Roland Widuch

Declaration of Christoph Picht

Declaration of David Birnbaum, Ph.D.

were caused to be served on the following attorney of record:

David L. McCombs
Haynes and Boone, LLP
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Tel: (214) 651-5533

Email: David.McCombs@haynesboone.com

By Email in lieu of First Class Mail (by consent)

/Guy Yonay/
Guy Yonay
Attorney for Patentee
Registration No. 52,388

Dated: January 10, 2011

Pearl Cohen Zedek Latzer, LLP
1500 Broadway, 12th Floor
New York, NY 10036
(646) 878-0800 (phone)
(646) 878-0801 (facsimile)

Appendix A

EXHIBIT A – SUPPORT FOR NEW CLAIMS

In connection with the response to the Office action issued November 19, 2010, Patentee has added new claims 54-82. The claims add no new matter, are supported by the specification of the '789 patent, and do not broaden the scope of the original patent.

Claim 54 finds support throughout the specification, including the following:

Claim 54	'789 patent specification
A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates to perform the following:	"The present method according to the invention can also be present implemented on a system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, the data processing unit being programmed in such a way that it operates in accordance with an embodiment of the method according to the invention." (Col. 8, lines 1-8)
(i) read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
(ii) analyze the input print data steam by means of a parser for graphically representable objects and split up the input print data stream into these graphically representable objects;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)
(iii) store said graphically representable objects in a memory in an object-oriented format;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)

Claim 54	'789 patent specification
(iv) transform the graphically representable objects into a format for the control of a printer; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer. . ." (Col. 3, lines 5-20)
(v) combine the objects into an output print data stream, and output said combined output print data stream to said printer,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)
wherein said graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)

Claim 55 finds support throughout the specification, including the following:

Claim 55	'789 patent specification
The system according to claim 54,	See above.
characterized in that the data processing unit is further programmed to send and receive e-mails in the cases defined in the script.	"It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver. . ." (Col. 5, lines 21-26) "Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails. . ." (Col. 5, lines 38-40)

Claim 56 finds support throughout the specification, including the following:

Claim 56	'789 patent specification
The system according to claim 54, further comprising:	See above.
characterized in that the data processing unit is further programmed to print original print and image data without a printer driver.	"Systems that operate on the method according to the invention, such as printing systems, are capable. . . of printing original print and image data without a printer driver." (Col. 5, lines 38-41)

Claim 57 finds support throughout the specification, including the following:

Claim 57	'789 patent specification
The system according to claim 54, further comprising:	See above.
an operating station with display means and input means, wherein said operating system makes it possible for the graphically representable objects to be read out via the application interface, to be changed, to be deleted, or to be appended before they are output in the output print data stream.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 58 finds support throughout the specification, including the following:

Claim 58	**'789 patent specification**
The system according to claim 57,	See above.
wherein said graphically representable objects to be read out, changed, deleted, or appended via the application interface are script objects.	"A further preferred embodiment of the method according to the present invention is characterized in that the graphically representable objects stored in the memory in an object-oriented format, preferably also script objects (for example Java Script objects), preferably before they are output in the output print data stream, are kept ready by an application interface to be read out, to be changed, to be deleted or to have new objects appended." (Col. 7, lines 9-16) "The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface, by assigning the methods required for this to the respective objects in accordance with their class hierarchy. This means that the objects stored in the memory can, for example, be displayed on a screen and modified as desired." (Col. 7, lines 24-32)

Claim 59 finds support throughout the specification, including the following:

Claim 59	'789 patent specification
A printer adapted for the transformation of digital print data streams comprising:	"The system according to the invention can also be integrated into a printer. . ." (Col. 8, lines 24-25)
an input to read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)
a memory to store said graphically representable objects in an object-oriented format,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)
wherein said printer is adapted to transform the graphically representable objects into a format for the control of an output device, to combine the objects into an output print data stream, and to output said combined output print data stream, and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer and the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)
wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in the cases defined in the script.	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)

Claim 60 finds support throughout the specification, including the following:

Claim 60	'789 patent specification
The system according to claim 59,	See above.
wherein said script is configured to operate as a timer, such that the script is executed automatically as a result of expiry of time.	"For example, the automatically occurring case, defined at least in the respective script, relating to the execution of the script can be configured as a timer, that is to say as a case which occurs automatically as a result of the expiry of a time, this timer preferably operating cyclically, that is to say starting itself again upon expiry." (Col. 6, lines 25-30).

Claim 61 finds support throughout the specification, including the following:

Claim 61	'789 patent specification
The system according to claim 59,	See above.
wherein said script when executed operates to obtain information over a network, and to use the obtained information in the output print data stream.	"Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails and of printing original print and image data without a printer driver. They are able to store any desired information on hard disks or memory cards and make said data available to all the devices connected in the network and Internet." (Col. 5, lines 38-44) "A further preferred embodiment of the method according to the invention is characterized in that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails. (Col. 5, lines 11-18)"

Claim 62 finds support throughout the specification, including the following:

Claim 62	**'789 patent specification**
The system according to claim 59,	See above.
wherein said printer is adapted to print original print and image data without a printer driver.	"Systems that operate on the method according to the invention, such as printing systems, are capable. . . of printing original print and image data without a printer driver." (Col. 5, lines 38-41)

Claim 63 finds support throughout the specification, including the following:

Claim 63	**'789 patent specification**
The system according to claim 59, further comprising:	See above.
wherein said printer is further adapted to send and receive e-mails in the cases defined in the script.	"It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver. . ." (Col. 5, lines 21-26) "Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails. . ." (Col. 5, lines 38-40)

Claim 64 finds support throughout the specification, including the following:

Claim 64	'789 patent specification
A printing system comprising	Throughout application.
a printer adapted for the transformation of digital print data streams comprising:	"The present invention relates to a method and system for the transformation of digital print data streams and corresponding printer and printer server." (Col. 1, lines 6-9)
an input to read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)
a memory to store said graphically representable objects in an object-oriented format,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)
wherein said printer is adapted to transform the graphically representable objects for the control of an output device, to combine the objects into an output print data stream, and to output said combined output print data stream, and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer and the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)

Claim 64	**'789 patent specification**
wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in the cases defined in the script; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)
an operating station with display means and input means, wherein said operating station is adapted to allow manipulation of said at least one script via an application interface.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 65 finds support throughout the specification, including the following:

Claim 65	'789 patent specification
The printing system according to claim 64, wherein said manipulation comprises at least one operation selected from the operations consisting of: reading out, changing, deleting, or appending said script.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 66 finds support throughout the specification, including the following:

Claim 66	'789 patent specification
The printing system according to claim 64, wherein said printer is adapted to print original print and image data without a printer driver.	"Systems that operate on the method according to the invention, such as printing systems, are capable. . . of printing original print and image data without a printer driver." (Col. 5, lines 38-41)

Claim 67 finds support throughout the specification, including the following:

Claim 67	'789 patent specification
The printing system according to claim 64, wherein said printer is further adapted to send and receive e-mails in the cases defined in the script.	"It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver. . ." (Col. 5, lines 21-26) "Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails. . ." (Col. 5, lines 38-40)

Claim 68 finds support throughout the specification, including the following:

Claim 68	'789 patent specification
A printer server for the transformation of digital print data streams comprising:	"The present invention relates to a method and system for the transformation of digital print data streams and corresponding printer and printer server." (Col. 1, lines 6-9)
an input to read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)

Claim 68	'789 patent specification
a memory to store said graphically representable objects in an object-oriented format,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)
wherein said printer server is adapted to transform the graphically representable objects for the control of an output device, to combine the objects into an output print data stream, and to output said combined output print data stream to the output device, and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer and the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)
wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in the cases defined in the script.	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)

Claim 69 finds support throughout the specification, including the following:

Claim 69	'789 patent specification
The printer server of claim 68, wherein said script is configured to operate as a timer, such that the script is executed automatically as a result of expiry of time.	"For example, the automatically occurring case, defined at least in the respective script, relating to the execution of the script can be configured as a timer, that is to say as a case which occurs automatically as a result of the expiry of a time, this timer preferably operating cyclically, that is to say starting itself again upon expiry." (Col. 6, lines 25-30).

Claim 70 finds support throughout the specification, including the following:

Claim 70	'789 patent specification
The printer server of claim 68, wherein said script when executed operates to obtain information over a network, and to use the obtained information in the output print data stream.	"Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails and of printing original print and image data without a printer driver. They are able to store any desired information on hard disks or memory cards and make said data available to all the devices connected in the network and Internet." (Col. 5, lines 38-44) "A further preferred embodiment of the method according to the invention is characterized in that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails. (Col. 5, lines 11-18)"

Claim 71 finds support throughout the specification, including the following:

Claim 71	'789 patent specification
The printer server of claim 68, wherein said output device is a printer.	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer[.]" (Col. 3, lines 5-14)

Claim 72 finds support throughout the specification, including the following:

Claim 72	**'789 patent specification**
A printing system comprising	Throughout application.
a printer server adapted for the transformation of digital print data streams comprising:	"The present invention relates to a method and system for the transformation of digital print data streams and corresponding printer and printer server." (Col. 1, lines 6-9)
an input to read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)
a memory to store said graphically representable objects in an object-oriented format,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)
wherein said printer server is adapted to transform the graphically representable objects into a format for the control of a printer, to combine the objects into an output print data stream, and to output said combined output print data stream to the printer, and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer and the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)

Claim 72	'789 patent specification
wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in the cases defined in the script; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)
an operating station with display means and input means, wherein said operating station is adapted to allow manipulation of said at least one script via an application interface.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 73 finds support throughout the specification, including the following:

Claim 73	**'789 patent specification**
The printing system according to claim 72, wherein said manipulation comprises at least one operation selected from the operations consisting of: reading out, changing, deleting, or appending, said script.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 74 finds support throughout the specification, including the following:

Claim 74	**'789 patent specification**
The printing system according to claim 72, wherein said printer server is adapted to print original print and image data on the printer without a printer driver associated with the printer.	"Systems that operate on the method according to the invention, such as printing systems, are capable. . . of printing original print and image data without a printer driver." (Col. 5, lines 38-41)

Claim 75 finds support throughout the specification, including the following:

Claim 75	'789 patent specification
The system according to claim 72, wherein said printer server is further adapted to send and receive e-mails in the cases defined in the script.	"It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver. . ." (Col. 5, lines 21-26) "Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails. . ." (Col. 5, lines 38-40)

Claim 76 finds support throughout the specification, including the following:

Claim 76	'789 patent specification
The method of claim 1, wherein said input data stream is formatted in a page description language.	"In this case. . . the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used." (Col. 3, lines 21-25)

Claim 77 finds support throughout the specification, including the following:

Claim 77	'789 patent specification
The method of claim 76, wherein said parser is a syntax analyzer, and wherein said analyzing by means of said parser comprises performing syntactic analysis on said input data stream.	"In this case. . . the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used." (Col. 3, lines 21-26)

Claim 78 finds support throughout the specification, including the following:

Claim 78	'789 patent specification
The method of claim 1, wherein storing the graphically representable objects in said object-oriented format comprising storing said graphically representable objects based on membership in hierarchically organized classes.	"The individual graphic objects are stored by using their membership of specific – expediently suitably hierarchically organized – classes. . . " (Col. 3, lines 41-44)

Claim 79 finds support throughout the specification, including the following:

Claim 79	'789 patent specification
The method of claim 1, further comprising dynamically linking a plurality of objects.	"In relation to the above explanations, it should be noted that the embodiments of the method according to the invention which themselves provide other objects with objects, for example by forwarding them or keeping them ready to receive or for interrogation by a script, for example, are also covered by the term 'dynamic object linking' (DOL)." (Col. 5, lines 32-37)

Claim 80 finds support throughout the specification, including the following:

Claim 80	'789 patent specification
The method of claim 1, wherein the objects are managed by display list management module.	"The objects are preferably managed here by means of a "display list management", which supports one-page and multi-page documents at as many levels as desired and which can be expanded dynamically by new objects." (Col. 3, lines 38-41)

Claim 81 finds support throughout the specification, including the following:

Claim 81	'789 patent specification
The method of claim 80, wherein the display list management supports one page and multi-page documents at a plurality of levels.	"The objects are preferably managed here by means of a "display list management", which supports one-page and multi-page documents at as many levels as desired and which can be expanded dynamically by new objects." (Col. 3, lines 38-41)

Claim 82 finds support throughout the specification, including the following:

Claim 82	'789 patent specification
The method of claim 81, wherein the display list management can be expanded dynamically by new objects.	"The objects are preferably managed here by means of a "display list management", which supports one-page and multi-page documents at as many levels as desired and which can be expanded dynamically by new objects." (Col. 3, lines 38-41)

Appendix B

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF NEW JERSEY

CCP SYSTEMS AG, Plaintiff, v. SAMSUNG ELECTRONICS CORP., LTD., SAMSUNG ELECTRONICS AMERICA, INC., SAMSUNG NETWORKS, INC. and IBM CORPORATION, Defendants.	Civil Action No: Jury Trial Demanded

VERIFIED COMPLAINT FOR COPYRIGHT AND PATENT INFRINGEMENT

CCP Systems AG ("CCP") for its Verified Complaint against Samsung Electronics Corp., Ltd. ("Samsung ECL"), Samsung Electronics America, Inc. ("Samsung America"), Samsung Networks, Inc. ("Samsung Networks"), and IBM Corp. ("IBM") (collectively, "Defendants"), states as follows:

PARTIES

1. CCP is a corporation organized and existing under the laws of Germany with its principal place of business at Stammheimer Str. 35, 70435 Stuttgart, Germany.

2. Upon information and belief, Defendant Samsung ECL is a corporation organized and existing under the laws of South Korea, with its principal place of business at Samsung Main Bldg. 250, 2-Ga, Taepyung-Ro Joong-Gu, Seoul, Korea, 100742.

3. Upon information and belief, Defendant Samsung America is a corporation organized and existing under the laws of the State of New York, with its principal place of business at 105 Challenger Road, Ridgefield Park, New Jersey, 07660.

4. Upon information and belief, Defendant Samsung Networks is a corporation organized and existing under the laws of South Korea, with its principal place of business at 9th Floor, Asem Tower, Samseong 1(il)-Dong , Gangnam-Gu Seoul, Korea, 135798.

5. Upon information and belief, Defendant IBM is a corporation organized and existing under the laws of the State of New York, with its principal place of business at 1 New Orchard Road, Armonk, NY 10504.

JURISDICTION AND VENUE

6. This is an action for copyright infringement under the Copyright Act, 17 U.S.C. § 106, and for patent infringement under the Patent Act, 35 U.S.C. §§ 271 and 281-285.

7. This Court has subject matter jurisdiction over the Federal law claims in this case pursuant to 28 U.S.C. §§ 1331 and 1338(a)-(b).

8. Venue is proper in this District pursuant to 28 U.S.C. §§ 1391 and 1400 because the Defendants may be found in this District, do substantial business in this District, and have engaged in acts of copyright and patent infringement in this District.

9. This Court has personal jurisdiction over Defendant Samsung America because New Jersey is Samsung America's principal place of business.

10. This Court has personal jurisdiction over Defendant IBM by virtue of IBM's transacting business within this State, and because of its systematic and continuous contacts with residents of this State.

11. This Court has personal jurisdiction over Defendants Samsung ECL and Samsung Networks by reason of those Defendants transacting business within this State and committing wrongful acts within this State, and by virtue of their systematic contacts with residents of this State.

12. In particular, this Court has personal jurisdiction over Defendants Samsung ECL and Samsung Networks because they operate and maintain interactive websites which can be accessed, and, upon information and belief, have been accessed, by residents of this State.

FACTS COMMON TO ALL CLAIMS

A. CCP's Software Products

13. CCP designs, develops, manufactures, sells and distributes software for use in connection with computer printers and other computer-related devices.

14. In particular, and relevant to this action, CCP developed software called "JScribe®," which is an open application and communication platform for servers, workstations, multi-function printers, and standard printers, allowing those devices to exchange information proactively in every direction.

15. The JScribe software has been the basis for many additional programs and variations. CCP has designed and developed those additional programs and variations into several applications, including the following:

a. "JScribe Core," which is the version of JScribe for embedding on individual printers, multi-function printers, and other devices.

b. "JScribe Mobile Print Solution" ("JMPS"), which is a JScribe-based client/server solution for mobile printing in company and public networks.

c. "JScribe Software Developer Kit" ("JSDK"), which is a JScribe-based integrated development environment for the development of JScribe applications.

d. "JISS OpenPower," which stands for "JScribe Intelligence Server Solution OpenPower," which is a server-based solution for tracking print and managing related costs, implemented on Linux/Unix platforms according to requirements defined by IBM.

f. "JTalk" is a JScribe-based tool for deploying JScribe applications to printers and other devices with JScribe Core or the JScribe Application Server Solution ("JASS") embedded.

g. "KYAOC" is a custom derivative of JMPS developed by CCP for a specific client, the State of Kentucky Administrative Office of Courts.

(The JISS OpenPower, JMPS, JSDK, JScribe Core, JTalk and KYAOC software are referenced collectively throughout this Complaint as the "CCP Software.")

B. <u>The Limited Licenses Granted to Defendants by CCP</u>

16. In 2004, CCP signed a contract with IBM Deutschland GmbH ("IBM Germany") (the "IBM Germany Agreement"), under which, *inter alia*, (a) CCP would license and deliver copies of JScribe Core to IBM Germany, and (b) IBM Germany then would sublicense and deliver copies of the JScribe Core software to its customers, solely to be bundled with other software in the customer's "Firmware" and embedded into the customer's device, such as a printer. ("Firmware" is a microprogram stored in ROM [read-only-memory], designed to implement a function that had previously been provided in software.) Under the IBM Germany Agreement, no other uses of the JScribe Core software were permitted by IBM Germany or its customers.

17. Upon information and belief, Samsung ECL, under a written sublicense agreement with IBM Korea (IBM Germany's Korean sister company, which, upon information and belief, has a sublicense arrangement with IBM Germany), embedded the JScribe Core software with other software of Samsung ECL, creating "Firmware," which it would then in turn embed directly into a Samsung-brand device, such as a printer.

18. According to the IBM Germany Agreement, as an IBM customer with an authorized sublicense, Samsung ECL, for itself and through its affiliates, such as Samsung America, could distribute devices containing the Firmware (which included the JScribe Core) to consumers in

the United States and elsewhere. Samsung would then pay a royalty to IBM Germany (or its sister company in Korea), who would then in turn pay a royalty to CCP for each device sold that incorporated the JScribe Core software.

19. The IBM Germany Agreement did not provide that an IBM customer, such as Samsung ECL, could distribute the JScribe Core software (or any other CCP Software) apart from a device, or to make any CCP Software publicly available online.

C. The IBM Agreement is Terminated

20. The IBM Germany Agreement, as amended, allowed CCP to terminate that agreement without further notice if certain events transpired. In particular, the IBM Germany Agreement allowed CCP to terminate the Agreement if CCP and IBM Germany could not reach agreement as to CCP's proposed business relationship with a competitor of IBM Germany. Since CCP and IBM Germany were not able to reach resolution on a proposed relationship between CCP and Samsung (a competitor of IBM Germany), CCP, on May 25, 2009 (the "Termination Date"), sent notice to IBM Germany that as of the Termination Date, CCP was terminating the IBM Germany Agreement and the license to the CCP Software granted therein.

21. Samsung ECL acknowledged the termination of the IBM Germany Agreement in an email dated July 15, 2009. In that email, Mr. Chin Yoon, the Vice President of Samsung ECL's Solution Business Group, instructed Samsung ECL personnel to refrain from marketing devices incorporating CCP Software, and notified them that the agreement under which the CCP Software was supplied to Samsung ECL had been cancelled.

22. Although CCP terminated the IBM Germany Agreement and the license therein to CCP Software as of the Termination Date of May 25, 2009, the Defendants, without CCP's consent,

have continued to reproduce and publicly distribute the CCP software, both as stand-alone software (bundled into Firmware) and in devices.

23. In addition, without CCP's consent, Samsung ECL, Samsung Networks, and Samsung America have placed the CCP Software, including the Firmware incorporating JScribe Core software, online at Samsung Network's website "downloadcenter.samsung.com" and "samsungprinter.info," available for free to the public.

COUNT I - COPYRIGHT INFRINGEMENT: DEFENDANT SAMSUNG NETWORKS

24. CCP incorporates the allegations of paragraphs 1 through 23 above as though fully set forth herein.

25. Samsung Networks owns and operates the website samsung.com. The samsung.com website has many sub-domains, including downloadcenter.samsung.com (the "Download Center Website").

26. Any United States resident with an Internet connection can access the Download Center Website, so long as that person knows the URLs for the code files on that website; no password or other security key is required to access it.

27. Samsung Networks has reproduced, publicly displayed, and, upon information and belief, publicly distributed the JScribe Core software on the Download Center Website, as part of the Firmware, and this conduct continues as of the date this Complaint was filed.

28. Anyone, including New Jersey residents, can download the Firmware, incorporating the JScribe Core software, for free, from the Download Center Website, if the person has the URL links to the Firmware files on the Download Center Website.

29. CCP did not authorize Samsung Networks or anyone else to make its JScribe Core software publicly available online.

30. CCP developed the JScribe Core software over the course of ten years, and that software program is an original work of authorship.

31. CCP has registered claims to copyrights in four versions of the JScribe Core software (JScribe Core v. 4.0, JScribe Core v. 4.1, JScribe Core v. 4.2, and JScribe Core v. 4.3) with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-843 to the JScribe Core v 4.0 claim to copyright, registration number TXu-1-610-827 to the JScribe Core v 4.1 claim to copyright, registration number TXu-1-610-915 to the JScribe Core v 4.2 claim to copyright, and registration number TXu-1-610-918 to the JScribe Core v 4.3 claim to copyright. These four versions will be referenced collectively throughout this Complaint as "JScribe Core."

32. By reproducing and publicly displaying the JScribe Core software at the Download Center Website, and by publicly distributing it on that site, Samsung Networks has directly infringed CCP copyrights in the JScribe Core software.

33. Samsung Networks knew that i) making the JScribe Core software available online, apart from devices, was not authorized under any agreement with CCP; and ii) in any event, as of May 25, 2009, all licenses to the JScribe Core software had terminated. Therefore, Samsung Networks' making the JScribe Core software available online for free download constitutes a willful infringement of CCP's copyrights.

34. CCP has suffered irreparable harm because of Samsung Networks' infringement of its copyrights in the JScribe Core software, and will continue to suffer irreparable harm in the future unless the Defendants are enjoined from infringing CCP's copyrights in the JScribe Core software.

35. CCP has suffered damages as a result of Samsung Networks' infringing conduct.

WHEREFORE, CCP requests the following relief against Samsung Networks:

a) an award of compensatory damages and disgorgement of any profits of Samsung Networks attributable to the infringement, along with prejudgment interests and costs;

b) a preliminary and permanent injunction prohibiting Samsung Networks and its officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing CCP's copyrights in the JScribe Core software, specifically prohibiting Samsung Networks from i) making the JScribe Core software available on the Download Center Website, ii) displaying or distributing any links to the Download Center Website; and iii) otherwise reproducing, displaying, distributing or preparing derivatives of the CCP Software;

c) a Court order requiring Samsung Networks to impound any infringing articles in its possession, custody, or control; and

d) such other relief as this Court deems just and proper.

COUNT II - CONTRIBUTORY AND VICARIOUS COPYRIGHT INFRINGEMENT: DEFENDANTS SAMSUNG ECL AND SAMSUNG AMERICA

36. CCP incorporates the allegations of paragraphs 1 through 35 above as though fully set forth herein.

37. Samsung ECL owns and operates a website at the URL samsungelectronics.com. That website has many sub-domains, including the sub-domain ecms.samsungelectronics.com (the "Partner Site").

38. Upon information and belief, Samsung ECL operates the Partner Site for the benefit of its dealers, developers and distributors worldwide, including in New Jersey.

39. To access content on the Partner Site, one must be registered with Samsung ECL, and Samsung ECL must approve, each application for access to the Partner Site.

40. Both before and after May 25, 2009, Samsung ECL made a spreadsheet available on the Partner Site. That spreadsheet contains information about various Samsung ECL products, including the Firmware that incorporates CCP's JScribe Core software.

41. That spreadsheet contains active hyperlinks to the Download Center Website. When one clicks on a hyperlink in the spreadsheet, one is taken directly to a ZIP file on the Download Center Website, where one can freely access and download the Firmware containing CCP's JScribe Core software.

42. The spreadsheet described in paragraphs 40 and 41 is downloadable and transferable. That is, anyone accessing that spreadsheet on the Partner Site can save that spreadsheet to his computer, copy it, attach it to an email, forward it to others, etc.

43. Samsung America controlled the website samsungprinter.info (the "Samsung Printer Website"). The Samsung Printer Website was open to the public.

44. Samsung America placed a spreadsheet similar to the one described in paragraphs 41 and 42 above on the Samsung Printer Website, containing links to the code files on the Download Center Website.

45. No password or other security information was required to access the code files on the Download Center Website, so anyone with this spreadsheet could freely access the code files in the Download Center Website.

46. The unauthorized availability of JScribe Core on the websites described above materially impacts the market for this software and CCP's ability to exploit its product. As an example of the effect on the market for the Firmware, CCP discovered an inquiry on a developer chat website, "Fix Ya," attached as Exhibit A. In that post, a web user asked whether anyone knew where to find the Firmware for a Samsung-brand printer with JScribe 4.0 embedded. Another user replied, identifying himself as a developer, and stated that if the user could not acquire the Firmware on the samsung.com website, he would get the Firmware from the Partner Site and send it to him. This example illustrates how Samsung ECL's, Samsung America's, and Samsung Networks' posting of the Firmware code files online has resulted in uncontrolled distribution of CCP's software products.

47. By posting the spreadsheet containing links to the Download Center Website online at the Partner Site and the Samsung Printer Website, Samsung ECL and Samsung America were aware of and supervised, facilitated and induced the direct infringement by Samsung Networks alleged in Count I. Further, by providing the means by which that infringement can occur, *inter alia*, Samsung ECL and Samsung America substantially participated in Samsung Networks' direct infringement. Accordingly, Samsung ECL and Samsung America contributorily infringed CCP's copyrights in the JScribe Core software.

48. In addition, by facilitating Samsung Networks' direct infringement, Samsung ECL and Samsung America garner goodwill with their partners, developers and distributors by making various products available on the Download Center Website and the Samsung Printer Website, and they also gain financial benefit by avoiding paying a royalty for use of the CCP Software. Accordingly, Samsung ECL and Samsung America have vicariously infringed CCP's copyrights in the JScribe Core software through their supervision and control of Samsung Networks.

49. CCP has suffered damages as a result of the infringing conduct of Samsung ECL and Samsung America.

50. CCP has suffered irreparable harm because of Samsung ECL's and Samsung America's contributory and vicarious infringement of its copyrights in the JScribe Core software, and will continue to suffer irreparable harm in the future unless Samsung ECL and Samsung America are enjoined from infringing CCP's copyrights in the JScribe Core software.

WHEREFORE, CCP requests the following relief against Samsung ECL and Samsung America:

a) an award of compensatory damages and disgorgement of any profits of Samsung ECL and Samsung America attributable to the infringement, along with prejudgment interests and costs;

b) a preliminary and permanent injunction prohibiting Samsung ECL, Samsung America, and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them from infringing CCP's copyrights in the JScribe Core software, specifically, prohibiting Samsung ECL and Samsung America from i) making the JScribe Core software available on the Download Center Website, ii) displaying or distributing any links to the Download Center Website; and iii) otherwise reproducing, displaying, distributing or preparing derivatives of the CCP Software;

c) a permanent injunction prohibiting Samsung ECL, Samsung America, and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and

all those controlled by or acting in concert with or in privity with any of them from infringing CCP's copyrights in the JScribe Core software, specifically, prohibiting Samsung ECL and Samsung America from i) making links to the CCP Software on the Download Center Website available on other websites such as the Samsung Printer Website, and ii) otherwise reproducing, displaying, distributing or preparing derivatives of the CCP Software;

d) a Court order requiring Samsung ECL and Samsung America to impound any infringing articles in their possession, custody, or control; and

e) such other relief as this Court deems just and proper.

COUNT III - COPYRIGHT INFRINGEMENT (DIRECT, CONTRIBUTORY AND VICARIOUS): DEFENDANT SAMSUNG AMERICA

51. CCP incorporates the allegations of paragraphs 1 through 50 as though fully set forth herein.

52. CCP has registered a claim to copyright in the JMPS software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-843 to the JMPS claim to copyright.

53. CCP has registered a claim to copyright in the JSDK software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-846 to the JSDK claim to copyright.

54. CCP has registered a claim to copyright in the JTalk software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-862 to the JMPS claim to copyright.

55. CCP has registered a claim to copyright in the KYAOC software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-844 to the JSDK claim to copyright.

56. Upon knowledge and belief, Samsung America placed copies of the CCP Software, including JScribe Core, JMPS, JSDK, JTalk and KYAOC, on the Samsung Printer Website, available for free download. In particular, Samsung America placed source code files for KYAOC on the Samsung Printer Website without authorization.

57. In addition, the Samsung Printer Website contained files with detailed instructions for how to use the CCP Software found on that website to install CCP Software onto hardware and other devices. These detailed instructions facilitated and encouraged unauthorized reproduction of the CCP Software by third parties, for which Samsung America garnered profit, thus, making those instructions available on the Samsung Printer Website constitutes contributory and vicarious infringement.

58. CCP never consented to or authorized the reproduction and distribution of its software products on the Samsung Printer Website.

59. CCP has suffered damages as a result of this infringing conduct.

60. CCP has suffered irreparable harm because of Samsung America's infringement of its copyright in the CCP Software, and will continue to suffer irreparable harm in the future unless Samsung America are enjoined from infringing CCP's copyrights in the CCP Software.

WHEREFORE, CCP requests the following relief against Samsung America:

a) an award of compensatory damages and disgorgement of any profits of the Defendants attributable to the infringement, along with prejudgment interests and costs;

b) a permanent injunction prohibiting Defendant Samsung America and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing CCP's copyrights in the CCP Software, specifically, by prohibiting Defendants from reproducing and distributing the CCP Software on other websites, such as the Samsung Printer Website, and by otherwise reproducing, distributing, displaying or preparing derivatives of the CCP Software;

c) a Court order requiring Samsung America to impound any infringing articles in its possession, custody, or control; and

d) such other relief as this Court deems just and proper.

COUNT IV - COPYRIGHT INFRINGEMENT: DEFENDANTS SAMSUNG ECL AND SAMSUNG AMERICA

61. CCP incorporates the allegations of paragraphs 1 through 60 as though fully set forth herein.

62. Samsung ECL reproduced CCP Software, namely, the JScribe Core software, in the process of creating the Firmware. Although the parties' agreements contemplated distribution of the Firmware (and with it, the CCP software) only embedded within a printer or other device, Samsung ECL bundled the Firmware for independent distribution on portable media, such as CDs or external hard drives, as well as by email distribution in code files. This bundling involved reproduction of CCP's JScribe Core software.

63. Once bundled and uploaded to such portable media or prepared for email distribution, Samsung ECL distributed that Firmware (including the JScribe Core software) to customers in the United States, through its US affiliate Samsung America in New Jersey.

64. Samsung ECL and Samsung America distributed some Firmware to customers who had already purchased a device containing an earlier version of the JScribe Core software. This Complaint will refer to this delivery as a "field upgrade."

65. Samsung ECL and Samsung America also distributed the Firmware to customers who had already purchased a device that did not contain any version of the JScribe Core software. This Complaint will refer to this delivery as a "field installation."

66. With each "field installation" and "field upgrade," Samsung ECL and Samsung America provided the customer with detailed instructions showing the customer how to install the Firmware onto an existing Samsung-brand device. These instructions effectively allow a customer to obtain the JScribe functionality without purchasing a JScribe-embedded device. Upon knowledge and belief, Samsung America and Samsung ECL have performed field installations and field upgrades for customers.

67. In addition to directly distributing the Firmware to customers, Samsung ECL and Samsung America made the Firmware (including the JScribe Core software) available on the Download Center Website and the Samsung Printer Website, along with detailed instructions showing a customer (or other web user) how to install the Firmware onto a Samsung-brand printer to achieve the functionality of a Samsung-brand printer with JScribe Core embedded.

68. CCP never consented to or authorized the reproduction and distribution of its software products as "field installations" and "field upgrades."

69. CCP has suffered damages as a result of this infringing conduct.

70. CCP has suffered irreparable harm because of Samsung ECL's and Samsung America's infringement of its copyright in the JScribe Core software, and will continue to suffer irreparable harm in the future unless Samsung ECL and Samsung America are enjoined from infringing CCP's copyright in the JScribe Core software.

WHEREFORE, CCP requests the following relief against Samsung ECL and Samsung America:

a) an award of compensatory damages and disgorgement of any profits of the Defendants attributable to the infringement, along with prejudgment interests and costs;

b) a permanent injunction prohibiting Defendants Samsung ECL and Samsung America and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing CCP's copyrights in the JScribe Core software, specifically, by prohibiting these Defendants from reproducing and distributing the JScribe Core software via the "field installations" and "field upgrades" as described herein, and by otherwise reproducing, distributing, displaying or preparing derivatives of the CCP Software;

c) a Court order requiring said Defendants to impound any infringing articles in their possession, custody, or control; and

d) such other relief as this Court deems just and proper.

COUNT V - COPYRIGHT INFRINGEMENT: DEFENDANT IBM

71. CCP incorporates the allegations of paragraphs 1 through 70 above as though fully set forth herein.

72. CCP has registered a claim to copyright in the JISS OpenPower software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-847 to the JISS OpenPower claim to copyright.

73. As noted above, CCP terminated the IBM Germany Agreement on May 25, 2009. Nevertheless, prior to May 25, 2009, and at least as early as December, 2006, the license to the JISS OpenPower software in the IBM Germany Agreement had expired by its own terms.

74. Upon information and belief, IBM has reproduced the JISS OpenPower in the process of selling servers and other devices, after December of 2006, and in any event after May 25, 2009. This reproduction of JISS OpenPower constitutes infringement of CCP's copyrights in the JISS OpenPower software.

75. Since December 2006, IBM has continued to market and, upon information and belief, sell products incorporating the JISS OpenPower software. For example, price lists and descriptions of products incorporating JISS OpenPower are available (at least as of August 7, 2009) on IBM's website. See Exhibit C.

76. Upon information and belief, IBM has publicly distributed the JISS OpenPower software in the United States, after December 2006. This public distribution of JISS OpenPower constitutes infringement of CCP's copyrights in the JISS OpenPower software.

77. CCP has suffered damages as a result of this infringing conduct.

78. CCP has suffered irreparable harm by IBM's infringement of CCP's copyrights in the JISS OpenPower software, and will continue to suffer irreparable harm in the future unless IBM is enjoined from infringing CCP's copyrights in that work.

WHEREFORE, CCP requests the following relief against IBM:

a) an award of compensatory damages and disgorgement of any profits of IBM attributable to the infringement, along with prejudgment interests and costs;

b) permanent injunctive relief prohibiting IBM and its officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing CCP's copyrights in the JISS OpenPower software, specifically, by prohibiting IBM from reproducing and distributing the JISS OpenPower software in devices, and by otherwise reproducing, distributing, displaying or preparing derivatives of the JISS OpenPower software;

c) a Court order requiring IBM to impound any infringing articles in its possession, custody, or control; and

d) such other relief as this Court deems just and proper.

COUNT VI - PATENT INFRINGEMENT (U.S. PATENT NO. 6,684,789) AGAINST SAMSUNG AMERICA, SAMSUNG ECL, AND SAMSUNG NETWORKS

79. CCP incorporates the allegations of paragraphs 1 through 78 above as though fully set forth herein.

80. On February 3, 2004, the United States Patent and Trademark Office (USPTO) duly and legally issued U.S. Patent No. 6,684,789 ("the '789 Patent") entitled "Method and System for the Transformation of Digital Print Data Streams and Corresponding Printer and Printer Server," was duly and legally issued to CCP, as assignee of inventor Thomas Krautter (a CCP employee). A copy of the '789 Patent is attached hereto as Exhibit B.

81. Samsung ECL, Samsung America, and Samsung Networks have been and are infringing, inducing infringement and/or contributing to infringement of the '789 Patent in this District, and throughout the United States, by making, selling, offering for sale, and/or importing infringing devices, software and other technology covered by one or more claims of the '789 Patent, including at least Samsung ECL's printer devices incorporating the JScribe Core technology.

82. As a direct and proximate result of the Defendants' infringement of the '789 Patent, CCP has suffered and continues to sustain monetary damages.

83. CCP has been and continues to be irreparably harmed by the Defendants' infringement of the '789 Patent. On information and belief, the Defendants will continue to infringe unless such infringement is enjoined by this Court.

WHEREFORE, CCP requests the following relief against the Defendants:

a) an award of compensatory damages and disgorgement of any profits of the Defendants attributable to the infringement, along with prejudgment interests and costs;

b) a permanent injunction prohibiting Defendants and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing, inducing the

infringement and/or contributing to the infringement of the '789 Patent pursuant to 35 U.S.C. §283; and

c) such other relief as this Court deems just and proper.

DEMAND FOR JURY TRIAL

Plaintiff CCP demands a jury trial on all issues.

Dated: August 25, 2009

Respectfully submitted,

SONNENSCHEIN NATH & ROSENTHAL LLP

By: s/ Marc S. Friedman
Marc S. Friedman
101 JFK Parkway
Short Hills, NJ 07078

and

1221 Avenue of the Americas
New York, NY 10020

212.768.6700
Fax: 212.768.6800
mfriedman@sonnenschein.com
bdelfin@sonnenchein.com
rstroder@sonnenschein.com

Attorneys for Plaintiff CCP Systems AG

Of Counsel:
Benito Delfin, Jr.
Rebecca Stroder

VERIFICATION OF CHRISTOPH PICHT UNDER 28 U.S.C. § 1746

I have read the foregoing Complaint. I have personal knowledge of the truth of the allegations contained therein, except for those allegations made upon information and belief.. For those allegations made upon information and belief, I believe them to be true.

I declare, under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed ~~the 25th day o~~f August, 2009

Christoph Picht

Attorney Docket No.: P-74637-US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patentee(s):	KRAUTTER, Thomas	Examiner:	STEELMAN, Mary J.
Control No.:	95/001,398	Reexamination Filed:	July 16, 2010
Patent No.	6,684,789	Art Unit:	3992
Title:	METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER		

DECLARATION OF ROLAND WIDUCH

I, Roland Widuch, declare as follows:

1. I am Chief Executive Officer ("CEO") of CCP Systems AG ("CCP"). CCP owns U.S. Patent No. 6,684,789 ("the '789 patent") that is the subject of the current reexamination.

2. From 1968 to 1995 I was employed by IBM Deutschland GmbH ("IBM"), IBM Corporation's German subsidiary. When I retired at IBM my title was Direktor Geschäftsbereich Medien (Director of the Media Business).

3. From 2002-2005, I was Commissioner General of CCP. In 2005, I retired from CCP.

4. On November 2008, I returned to CCP as CEO, and have been CEO since that date.

5. I believe Samsung has sold at least approximately half a million printers in the United States, and many more than that amount on a worldwide basis, that include the JScribe® technology.

6. Samsung also allowed its customer and partners to download CCP's JScribe® technology from Samsung's website without permission from CCP. CCP does not know the full extent of Samsung's infringement, but believes there was a large number of downloads as well.

7. In 2003-2004, IBM evaluated CCP's technology, including the patent at issue. After satisfying itself of the value of this technology (and inherently its innovativeness), in 2004 IBM took an exclusive license to CCP's JScribe® software ("the IBM License"). The exact amounts at issue may be confidential to IBM, and CCP will not disclose them here. However, the agreement committed IBM to pay a minimum of tens of millions of dollars over the next five years. In 2005, IBM was given the right to sub-license the technology under certain conditions.

8. IBM's first sub-licensee under the IBM License, Konica-Minolta Business Solutions, Inc., ("Konica-Minolta"), paid IBM millions of dollars in a lump sum payment plus running royalties for the right to embed CCP's technology into its products. Under the terms of the IBM License, CCP received 50% of the payments made to IBM. Konica-Minolta, through IBM, paid CCP several million dollars for the right to embed CCP's technology into its products.

9. In February 2005, IBM and CCP entered into an agreement called a "Co-Marketing Logo License Agreement." A copy of Exhibit 2 to that Agreement is attached as Exhibit A hereto.

10. In about 2006, IBM, through its subsidiary in Korea, sublicensed the technology to Samsung Electronics Co. Ltd. ("Samsung"). Samsung paid IBM a lump sum in the millions of dollars, and was supposed to pay a running royalty for each device that included the licensed JScribe® technology. Under the terms of the IBM License CCP was supposed to get half of the lump sum payment and half of the running royalties.

11. Upon granting to Samsung a sublicense to the JScribe® technology, IBM and Samsung issued a joint press release that stated: "'JScribe', the printing optimized middleware of IBM will be launched in the Samsung digital printing devices such as printers and multifunction products." (Emphasis added). (See Exhibit B). It is unclear which of the two companies (or both) was responsible for drafting this document, but JScribe® has always been a CCP product. Nevertheless, IBM and/or Samsung took credit for the innovative technology.

12. In 2006, IBM awarded CCP its "Best Seller Award," a recognition of CCP's groundbreaking work on JScribe® technology. (See Exhibit C).

13. In January 2007, IBM sold the remainder of its printer business to Ricoh Company, Ltd. and terminated the exclusivity of the license. IBM continued to pay CCP some, but not all, of the royalties that IBM was obligated to pay under the IBM License as a result of Samsung's sales. CCP currently has litigation pending in Germany against IBM related to the amount of royalties that IBM was supposed to pay under the IBM License.

14. In June 2007, CCP's JScribe® product won the European Commission China Information and Communications Technologies Innovator Award (ChinICT). (See Exhibit D). CCP was selected from 60 enterprises as among the best innovators. (See Exhibit D).

15. In about 2008, Samsung issued a press release entitled: "New Samsung MFP with Advanced Features Offers Easy Integration for Efficient, Simple and Reliable Printing" that discussed the many advantages of the CCP product that Samsung characterized as "Samsung's JScribe™". (See Exhibit E). Samsung obviously does not own the trademark to JScribe, but it is interesting that Samsung appropriated this important technology as its own, technology that it now belittles.

16. In about 2008-09, Samsung praised the patented JScribe® technology effusive praise in a White Paper (See Exhibit F).

17. In my opinion, all the deals cited in the Samsung White Paper (¶ 15, Exhibit F) could only have been done because of the JScribe® technology in the Samsung devices.

18. Indeed, Samsung has described CCP's patented technology as follows: "JScribe is an embedded application and communication platform for MFPs [multifunctional printers]. The JScribe® software development Kit (SDK) allows third-parties the flexibility to shape and extend the functionality of Samsung's MFPs to meet their specific business requirements." (See Exhibit F, p. 1)

19. In 2009, CCP terminated the IBM License, and with it any rights that Samsung may have had under that agreement.

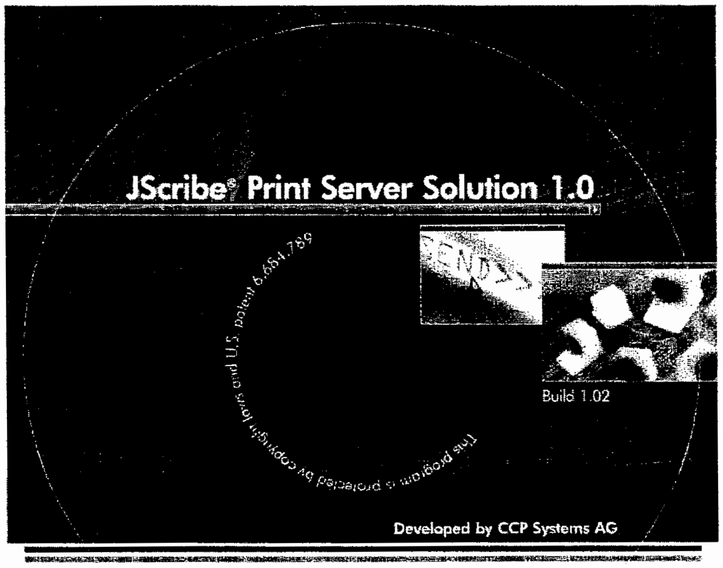
20. Even after having been sued, and after bringing on this reexamination, Samsung's web site continues to promote the advantages of "JScribe technology": "As printers have gone from only printing to complicated multi-function machines, Samsung has provided JScribe software to support all the new functions and features." (See Exhibit G).

I declare under penalty of perjury under the laws of the United States of America that all statements made here are based on my own knowledge and are believed to be true. I understand that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. section 1001), and may jeopardize the validity of the patent subject to reexamination.

Roland Widuch
Chief Executive Officer
CCP Systems AG

Jan, 03, 2011
Date

Exhibit A


JScribe® Print Server Solution 1.0
This program is protected by copyright laws and U.S. patent 6,684,789
Build 1.02
Developed by CCP Systems AG

IBM

Exhibit B

삼성전자, IBM 프린팅 제휴

◆ 삼성전자 박종우 사장(정면 오른쪽)과 한국IBM 이휘성 사장이 조인식 후 악수하고 있는 모습

삼성전자와 한국IBM이 엔터프라이즈 프린팅 사업 활성화를 위한 협정서를 체결하고 프린팅 시장에서 공동영업을 추진키로 했다. 두 회사는 이번 제휴로 국내 프린팅 시장을 위한 공동 비즈니스 모델을 발굴키로 했다. 특히 IBM의 기업용 프린팅 최적화 미들웨어 '제이스크라이브(JScribe)'를 프린터· 복합기 등 삼성의 디지털 프린팅 기기에 탑재할 계획이다. 삼성전자 박종우 사장은 "IBM의 기술과 삼성의 노하우를 결합해 기업용 프린팅 시장에서 더 큰 열매를 맺게 될 것" 이라고 말했다. 한국IBM 이휘성 사장도 "삼성전자와 협력을 통하여 보다 경쟁력 있는 제품으로 차별화된 비즈니스 가치를 제공할 수 있게 되었다"고 말했다.

Alliance in Printing, Samsung and IBM Korea

Samsung Electronics and IBM Korea have contracted an agreement for launching an enterprise printing business, and will be promoting a joint business in the printing marketplace. In the context of this alliance, the two companies have decided to establish a joint business model for the domestic printing market. In particular, 'JScribe', the printing optimized middleware of IBM will be included in the Samsung digital printing devices such as printers and multifunctional products. The President of Samsung Electronics, Park Jong-woo, said the combination of IBM technology and Samsung know-how will bring greater rewards in the enterprise printing market. "We will offer a differentiated business value with more competitive products in collaboration with Samsung Electronics," said Lee Hwee-sung, President of IBM Korea.

The photo shows Park Jong-woo, President of Samsung Electronics, and Lee Hwee-sung, President of IBM Korea, shaking hands at the signing ceremony.

Exhibit C

Pressemitteilung

IBM Preisverleihung an Business Partner

Gewinner des IBM Verkaufswettbewerbs und anderer Auszeichnungen auf IBM Channel Kick-off in Stuttgart geehrt

Stuttgart, 26. Januar 2006 – Volles Haus: Rund 700 IBM Business Partner folgten am 18. Januar der Einladung zum IBM Channel Kick-off nach Stuttgart, bei dem zahlreiche Preise für herausragende Leistungen der Partner verliehen wurden: IBM gab die Hauptgewinner des Wettbewerbs „Iron Business Man UNLIMITED 2005" bekannt und ehrte die Preisträger des „IBM BestSeller Awards 2005", des „Lotus Awards 2006" sowie des „Beacon Awards 2006".

„Die Zusammenarbeit mit unseren Partnern ist uns äußerst wichtig. In Deutschland gibt es über drei Millionen Unternehmen. Diesen Markt können wir als IBM gar nicht alleine abdecken," sagte Thomas Henkel, Vice President der IBM Business Partner Organisation in Deutschland und Gastgeber der Veranstaltung. „Wir wollen das Geschäft mit unseren Partnern weiter ausbauen und gemeinsam mit ihnen erfolgreich sein. Und dass IBM bereits über ein weitreichendes Partnernetzwerk verfügt, zeigt nicht zuletzt die rege Beteiligung an unseren Wettbewerben."

Beim ‚Iron Business Man UNLIMITED 2005' handelt es sich um einen IBM PartnerWorld Wettbewerb für Business Partner und Distributoren aus Deutschland, Österreich und der Schweiz, deren Verkäufe von Lenovo TopSeller Produkten, IBM SystemSeller Produkten und IBM Service-Angeboten gewertet werden. Der Hauptpreis für die 17 Gewinner ist eine Reise in die USA nach Las Vegas, New York und Los Angeles.

Der Wettbewerb fand vom 2. Mai bis 17. Dezember 2005 statt, die Preise wurden in drei Etappen vergeben: Nach der ersten fuhren 40 Partner aus Deutschland, Österreich und der Schweiz, die sich durch den Verkauf von IBM SystemSeller- und Lenovo TopSeller-Produkten verdient gemacht hatten, nach Istanbul zur Deutschen Tourenwagenmeisterschaft (DTM). Weitere 40 gingen im November auf Wüsten-Safari nach Ägypten. Zehn Sieger der Lenovo Sonderwertung dürfen sich außerdem auf die Winterolympiade in Turin freuen.

Pressemitteilung

Neben dem Iron Business Man wurden der deutsche "BestSeller Award 2005" sowie die internationalen Preise "Lotus Award 2006" für die beste Mittelstandslösung und der "Beacon Award 2006" in der Kategorie „Bester Berater und Systemintegrator für On Demand Business Solutions in der Region Europa, Naher Osten und Afrika“ verliehen.

Die Gewinner der einzelnen Kategorien:

Iron Business Man UNLIMITED 2005:

Deutschland – Lenovo Partner:
AnNo Text GmbH, Cyberport.de GmbH, C&P Network Consulting GmbH, MECO Medizinische Computersysteme GmbH

Österreich – Lenovo Partner:
ACP Holding GmbH

Deutschland – IBM Partner:
ALSO Deutschland GmbH, Bechtle Logistik & Service, HK Computerdienst GmbH, MetaComp GmbH Computer + Netzwerke, netcon EDV & Service GmbH, Tech Data GmbH & Co. OHG – Geschäftsbereich TD Midrange

Österreich – IBM Partner:
ABAX Informationstechnik GmbH, Fritz & Macziol Computersysteme und IT Dienstleistungen, ILS Consult GmbH

Schweiz - IBM Partner:
Cyber Network SA, DATALINE AG, Paninfo AG

BestSeller Award 2005:

Kategorie Service:
CCP Systems AG, ONVENTIS GmbH

Kategorie Partner Solution Center (PSC):

Pressemitteilung

PSC Süd: ACP IT Solutions AG, PSC Südwest: levigo systems gmbh, PSC Mitte: NCT GmbH, PSC Nordwest: BIT Bücker GmbH, PSC Nord: Netzlink Informationstechnik GmbH, PSC Ost: procilon IT-Logistics GmbH

Kategorie Wachstum:
COMPAREX Deutschland GmbH, becom Informationssysteme GmbH, FRITZ & MACZIOL Software- und Computervertrieb GmbH

Kategorie Beste multiple Lösung
PROFI Engineering Systems AG

Kategorie ISVs
active logistics GmbH, BISON Solutions GmbH, C.I.S. Cross Industrie Software AG

Kategorie Software
System Vertrieb Alexander GmbH

Lotus Award 2006

Beste Mittelstandslösung weltweit: Kumatronik Software GmbH

Beacon Award 2006

SYSDAT GmbH

IBM Business Partner Organisation (BPO)
IBM generiert rund ein Viertel ihres Gesamtumsatzes über ihr breites Business Partner-Netzwerk, wobei dieser Anteil kontinuierlich zunimmt. Weltweit sind es rund 90.000 Unternehmen aus den Bereichen Systems & Services, Entwicklung sowie Software, die in Zusammenarbeit mit IBM maßgeschneiderte IT-Lösungen anbieten. Das IBM PartnerWorld Marketing- und Unterstützungsprogramm soll den Business Partnern die Zusammenarbeit mit IBM noch einfacher machen: Die Partner erhalten über ein Webportal einen komfortablen, konsolidierten Zugang zu gemeinsamen Marketingaktionen, technischem Support und vergünstigten Schulungen sowie zu IBM Finanzierungsangeboten.
Weitere Informationen zur IBM Business Partner Organisation finden Sie unter: www.ibm.com/de/partners

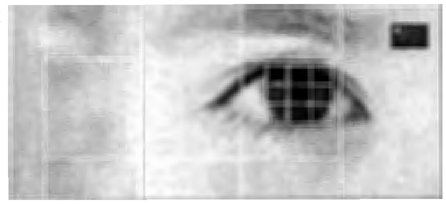
Weitere Informationen für Journalisten:
IBM Deutschland GmbH
Presse- und Öffentlichkeitsarbeit
Geschäftsbereich Mittelstand und Business Partner
Sabine Büttner
Telefon: (0711) 785-5898,
Telefax: (0711) 785-4528
E-Mail: Sabine_Buettner@de.ibm.com

Exhibit D

CHINICT 2007

Top ICT Innovators

CROWNE PLAZA PARKVIEW WUZHOU HOTEL BEIJING, P.R.CHINA.
MAY 24TH-25TH, 2007



Press release

THE JSCRIBE COMPANY

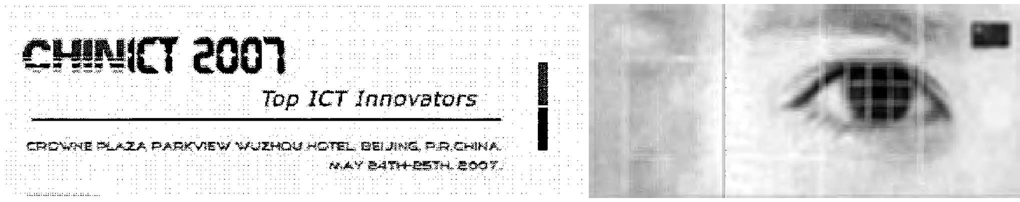
CCP Systems AG
Stammheimer Straße 35
70435 Stuttgart, Germany

CCP Systems wins the European Commission ChinICT Innovator Award

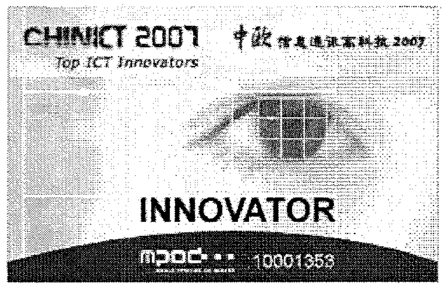
Beijing China / Stuttgart Germany, June, 2007 – On May 25th, CCP's technology – the open platform for applications and communication - won the European Commission ChinICT Innovator Award 2007, in Beijing.

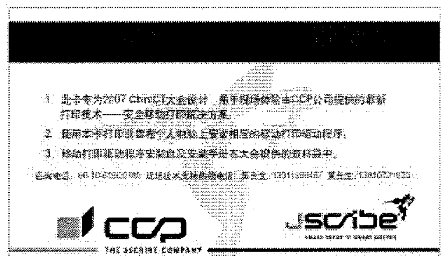
Under the high patronage of Viviane Reding, member of the European Commission responsible for Information Society & Media, and the municipality of Beijing the ChinICT editorial board short-listed 60 finalists considered to lead the next wave of innovations. The evaluation of these entrepreneurs happens throughout the EMEA and APAC region (Europe, Middle East and Africa and Asia Pacific). From these 60 enterprises the best among the best Innovators 2007 winners were selected.

"We are very proud to have been chosen from such a strong field of innovative companies" says Thomas Krautter, the founder and CTO of CCP Systems AG. "The ChinICT 2007 has been an excellent networking and business platform to introduce our JScribe technology. Winning this award validates our strategy to develop a worldwide unique and vendor independent technology platform for the complex, ambitious needs of the output management market. Individual performing applications on demand on the one hand and JScribe technology embedded in intelligent output devices on the other hand build the synergy of the future. "



mobile Printing on demand station at the ChinICT





mPod – empowered participant badge

ChinICT offered CCP the possibility to present the "mobile Printing on demand (mPod)" solution which is an example of the possibilities of JScribe. This mobile printing solution supports companies to integrated a secure "print follows user" solution into their IT environment and helps mobile workers to print on the fly wherever they want (HotSpots, hotels, air port launges, branches etc).

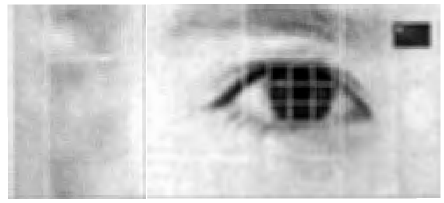
All participants of the ChinICT received a mPod empowered badge that allows (together with the installation of the CCP Universal Driver) to print their documents on demand at one of three mobile printing stations around the registration desk.

JScribe mobile Printing on demand solution means: print securely on the move. The CCP Universal Driver is at the heart of our mobile print solution, to ensure printing is easier and more flexible than ever before. You only need to install drivers for specific network printers once – on the central Secure Server. One "Universal Driver" on any workstation / notebook encrypts the print job – using a symmetrical public/private key encryption – before sending to the Secure Server. The print data stays on the server until the user identifies themselves at the printer or MFP – they can choose any device equipped with the JScribe mobile print solution. User identification can be by ID cards (like on the ChinICT) or biometrical data or a personal identification number. The user can then control the available print options on a dialogue unit connected to the printer or MFP. As soon as they choose to print, the server prepares the data, and sends it to the printer.

CHINICT 2007

Top ICT Innovators

CROWNE PLAZA PARKVIEW WUZHOU HOTEL BEIJING P.R.CHINA
MAY 24TH-25TH 2007



"CCP is honoured to have JScribe awarded as ChinICT Top Innovative technology," said Frank Lichner, CEO of CCP Systems AG. "With the highly flexible JScribe technology CCP develops consequently solutions for the market demands of today and the future. JScribe brings outstanding value to the customers in matters of filling the gap between document processes and output environment. Especially the secure and mobile print solutions will encourage the worldwide spread of mobile applications."

About ChinICT
ChinICT is a major event focused on the most innovative and fastest growing ICT SME (Information & Communication Technologies Small & Medium Enterprises) from Europe and China, taking place on May 24th & 25th, 2007 at the Crown Plaza Parkview Wuzhou Hotel, Beijing, P. R. China and leverages on the success of the 2006 edition. ChinICT benefits from outstanding media coverage and is strongly supported by companies such as Microsoft, Google, Huawei etc. The ChinICT annual awards recognize tomorrow's industry leaders by selecting promising technology companies who set new innovative standards.

About JScribe
JScribe technology offers the intelligent integration of devices in existing application infrastructures and enables the devices to communicate with each other. So the input and output hardware suddenly becomes an active part of business workflows. One enjoyable side-effect is the reducing network traffic and server loads by shifting the processing to "intelligent" printers and multifunction peripherals. Running the applications on printers or MFPs to simplify the specific printing and document management needs, will multiply the return on those investments. One more important point is that the workload and the costs of every output device can be detected and controlled.

About CCP Systems AG
CCP Systems AG develops technologies and solutions for managing and controlling output processes. More than 20 years of experience in manufacturing printer drivers and document solutions are the foundation for the patented JScribe® technology, a worldwide unique application and communication platform for digital printers and multifunctional devices.
JScribe is marketed worldwide by IBM and the deployment as a standard integrated technology embedded in devices by Konica Minolta and Samsung started in 2006.
CCP earned 5,6 Million Euro revenue with 35 permanent employees in 2006.

Exhibit E

SAMSUNG

NEW SAMSUNG MFP WITH ADVANCED FEATURES OFFERS EASY INTEGRATION FOR EFFICIENT, SIMPLE AND RELIABLE PRINTING

~ SCX-5835FN high-performance monochrome laser MFP expands Samsung's B2B product portfolio ~

Jan 2009 - Samsung Electronics, a leading name in the world of consumer electronics and information technology, today announced the launch of the SCX-5835FN monochrome laser multifunction printer (MFP). Designed for corporate and small-to-medium-business users, the SCX-5835FN delivers efficient, easy to use and reliable printing, thanks to new features including a 7" Colour Touch Screen and an 80GB hard disk drive (HDD). Despite its compact size and competitive price point, the SCX-5835FN offers the advanced features and solutions required to meet the demanding requirements of businesses.

The SCX-5835FN increases efficiency by delivering high speed printing of up to 33 pages per minute (ppm). It also processes documents quickly thanks to its powerful 360MHz processor and large 256MB memory capacity. This minimises the bottlenecks that can slow down workflow, making it ideal for the busy office environment. The SCX-5835FN is also network-ready, enabling users to easily send emails and share documents or images via the office network, without having to purchase additional devices. With the MFP's new intuitive 7" Colour Touch Screen, operating the device and sharing documents is quicker and simpler.

In order to improve productivity and reduce the cost of document management processes, the SCX-5835FN can be integrated with Samsung's exclusive SmarThru™ Workflow solution. This enables users to quickly and easily share digital documents by allowing them to scan, fax or email digital documents to multiple destinations across an organisation, from individual users and shared folders to FTP and HTTP sites. This reduces the amount of time taken to manage information in the organisation and eliminates costs associated with wasted time searching for documents.

With the SCX-5835FN, users can also be more efficient thanks to its enhanced toner yield of up to 10,000 pages and its ability to handle high volumes of paper (up to 1,100 sheets). This increased capacity means that users do not have to change the toner or reload the device with paper as frequently, saving time. Time can also be saved due to SCX-5835FN's direct USB interface, which means that users can print, scan and fax directly from their USB memory device without having to log on to a PC.

SAMSUNG

Corporate users can maximise their printing investment by customising the device to their specific requirements. For example, users can incorporate additional functionality such as security solutions or job accounting. Samsung's JScribe™ open platform technology means these new features can be added quickly and easily in response to the individual needs of the customer.

The new MFP boosts exceptional reliability. This, along with its high toner yield, maximum monthly duty cycle of 80,000 pages, and ability to handle high volumes of paper, make the SCX-5835FN well-suited to busy and demanding office environments.

"Enhancing our business product portfolio is a key to extending our leadership in the printing marketplace. With our advanced technology and focus on the business environment, our product family contains the right solution to meet the needs of any organisation," said Graham Long, Vice President, European Printing Operation, Samsung Electronics. "With this new device we have incorporated robust new features and offer faster print speeds than previous models, meeting customer requirements for efficiency in the workplace."

- End -

Technical specifications

Feature	SCX-5835FN
General	
Size (WxDxH)	500 x 465 x 547 mm
Weight (with consumables)	23 kg
Display	7' colour touch screen
System Memory	256MB RAM (expandable to 512MB)
Interface	USB 2.0
Maximum Monthly Duty Cycle	80,000 sheets
Toner Yield	10,000 pages (ISO/IEC19752)
Print	
Speed	Up to 33 ppm single-sided Up to 17 ppm double-sided (duplex)
Resolution	Optical 600 x 600 dpi; Enhanced 1200 x 1200 dpi
First Print Out Time	8.5 seconds
Emulation	PCL 6 / 5e; PS3
Copy	
Speed	33 copies per minute (cpm) single-sided 17 cpm double-sided (duplex)
Resolution (Optical)	600 x 600 dpi
First Copy Out Time	9.5 seconds
Zoom Rate	25% - 400 %
Multi Copy	1-999
Scan	

SAMSUNG

Feature	SCX-5835FN
Method	Colour CCD
Resolution (Optical)	600 dpi
Resolution (Enhanced)	4800 x 4800 dpi
Grey Scale	256
Paper Handling	
Input Capacity and Type	500-sheet Cassette Tray / 50-sheet Multipurpose tray
Output Capacity and Type	250-sheet Tray
Pricing and Availability [1]	
Pricing	Please contact a local Samsung dealer for pricing
Availability	Immediately

About Samsung Electronics
Samsung Electronics Co., Ltd. is a global leader in semiconductor, telecommunication, digital media and digital convergence technologies with 2007 consolidated sales of US$103.4 billion. Employing approximately 150,000 people in 134 offices in 62 countries, the company consists of four main business units: Digital Media Business, LCD Business, Semiconductor Business, and Telecommunication Business. Recognized as one of the fastest growing global brands, Samsung Electronics is a leading producer of digital TVs, memory chips, mobile phones and TFT-LCDs. For more information, please visit www.samsung.com.

[1] May differ from country to country

Exhibit F

JScribe

Samsung White Paper

Samsung
White Paper

Introduction

This White Paper describes JScribe technology for Samsung B2B applications. This document is intended for those who want to understand JScribe and its uses for Samsung B2B multifunctional devices (MFDs).

The role of a printer has changed significantly as the technology has matured. Printers were historically used for printing documents in support of a simple business requirement. This process would usually support a single user or a small group of users. And the printer was considered an accessory to the computer. Management of the printer and consumables was isolated to the individual or to the group without any visibility by the corporate IT infrastructure.

Today the demand by users for document features such as color printing, scanning, storage, tracking, and security, along with the needs of IT departments for usage metrics and remote management, have driven the development of complex multifunctional printers (MFPs) by Samsung. One obstacle to creating an MFP has been the complex embedded software that must be developed to support all of these multiple functions and features. JScribe is Samsung's solution to this obstacle.

JScribe is an embedded application and communication platform for MFPs, and it has been specifically designed to provide a custom application suite for these printers. The JScribe software development Kit (SDK) allows third-parties the flexibility to shape and extend the functionality of Samsung's MFPs to meet their specific business requirements.

The Four Layers of JScribe

An embedded JScribe system may be seen as 4 distinct layers:

The base firmware of the system, the JScribe Wrapper API, which exposes the system services to the JScribe Core, and JScribe applications written in JavaScript.

While the first 3 layers can be seen as fixed parts of the firmware, JScribe applications can also be dynamically uploaded to the MFP.

The JScribe Core

The JScribe core is written in strict ANSI-C and is the standard code basis for every platform. It has been designed to be highly flexible and portable to almost any microprocessor-based system. It is based on JavaScript 1.7, ECMA compliant, fully tested and serves as a robust basis by providing the complete application API for JScribe applications.

The core code is delivered as full source code to OEMs and there is no need to modify it.

The API

The API layer is the binding layer, which must translate the standardized function calls from the JScribe Core to the underlying firmware or operating system.

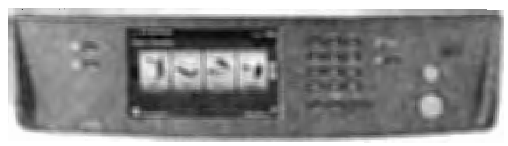
During embedding JScribe, the Wrapper API must be designed and adopted to a specific target. Once the initial embedding process has been finished, the platform is ready to run any JScribe application without modification.

Objects and Events

To allow enterprises to develop their own applications that leverage the full functionality of our MFPs, we allow the complete, object-oriented API to be available. Within JScribe there are several groups of objects available, which allow control over the MFP features. These objects are coded in ANSI-C and controlled by the JScribe application. This architecture guarantees a high overall performance of the complete system.

The following groups of service objects are available to a JScribe application:

- **User Interface**



User interface objects allow interaction between the user and the MFP control panel and keypad. JScribe has been designed to allow low level access to the control panel and keypad. By using the built-in JScribe panel functionality, developers can create custom applications for the control panel and keypad on the MFP.

- **Communication**

Communication objects allow an application to connect to and exchange any kind of data with any other JScribe platform or even non-JScribe devices. JScribe provides a straightforward and easy to use interface for performing this communication.

SAMSUNG

Samsung
White Paper
(Continued)

- **Connectivity**

Connectivity objects allow JScribe applications to control COM or USB port connected peripheral devices. A simple control script can serve as the device driver and can control a broad range of devices including:
- fingerprint readers
- Card readers
- RFID printers
- RFID readers
- label printers
- Slave printers.

- **Storage**

Storage objects allow JScribe applications to access the MFP's storage media, no matter what operating system or technology is used. This includes built-in flash memory, hard disks or USB drives.

- **Print Job Filter**

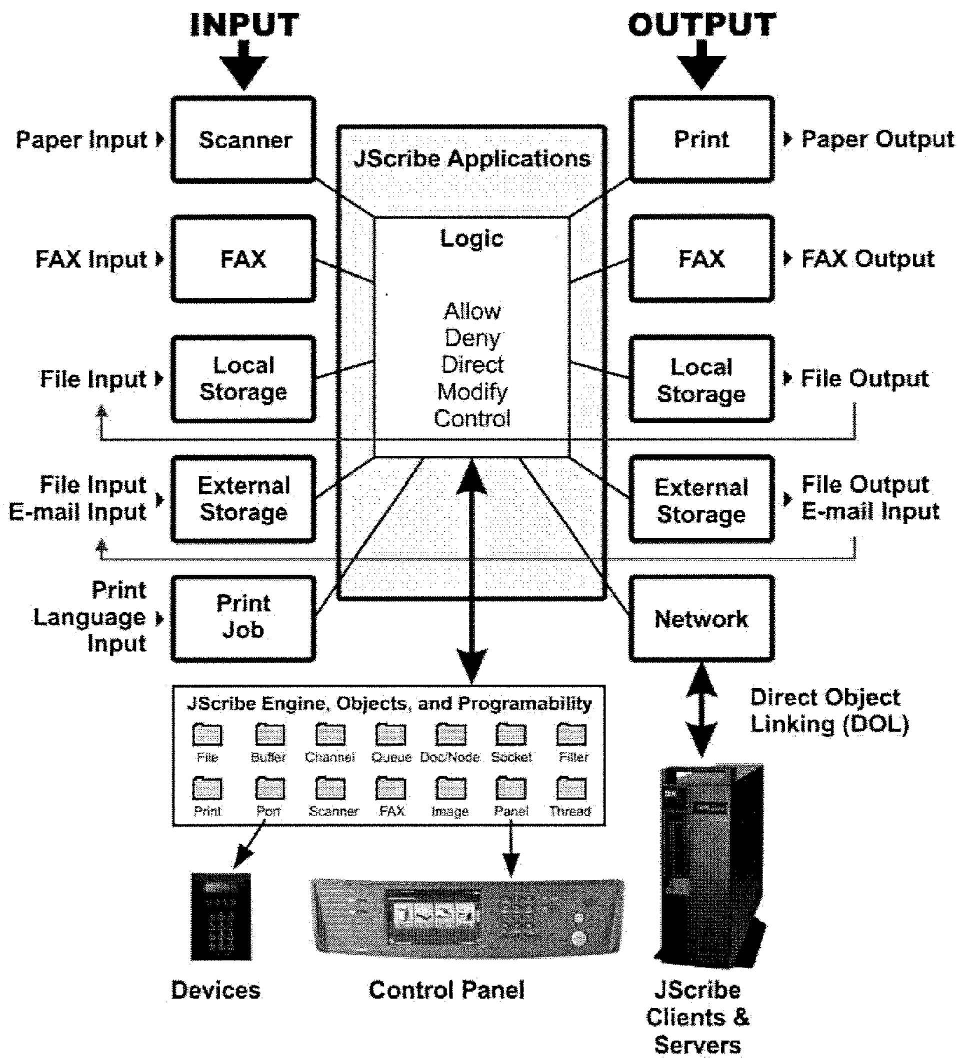
Print job filter objects allow JScribe applications to recognize incoming print data and, if required, modify the print sequences before the print data reaches the MFP's RIP. This solves all kinds of compatibility issues encountered when building variable data printing or electronic forms applications.

- **Security**

Security objects prevent the execution of unauthorized JScribe code. Administrators can set an MFP to ignore unauthorized executable code.

- **Multi-Functional Services**

JScribe also provides dedicated objects to control the scanner and FAX units of MFPs. These objects control the functionality of the scanner and the fax. This allows programmers to lock the scanner against unauthenticated users and to filtering out unwanted fax input.



Samsung
White Paper
(Continued)

JScribe Features

Most MFPs provide a set of built-in features, which allow users to perform complex tasks such as, scanning directly to Email or scanning directly to an FTP site. These built-in features are hardcoded into the printer. In the past, modifying the printer code was either impossible or extremely expensive. This left the user with very few choices when a new printer-related business task emerged. They could make expensive modifications to their existing MFPs, or they could make a capital investment in new MFPs, which could become obsolete with the discovery of the next new printer related business task.

Now JScribe allows users to easily customize and enhance JScribe enabled Samsung MFPs to meet their new business needs as they arise. The JScribe solutions are as easy to install as new software programs on a PC. And JScribe currently includes many MFP solutions, with new features being developed continually for Samsung.

Third party developers and integrators can also leverage the Samsung MFP JScribe solutions to meet custom implementations by using the JScribe SDK. With the SDK they can cover almost any functionality requested by customers. This functionality can include the following:
• Integration with other enterprise business processes
•Integration with other IT devices in a network environment
•Centralized development and deployment of new solutions
• Custom control panel designs
• User authorization
• Custom workflows
• Custom print processes
• ......

On the input side, JScribe allows you to access and control the MFP scanning and fax features. It also allows you to interact with storage devices such as built-in hard drives, and to monitor incoming network requests or print data.

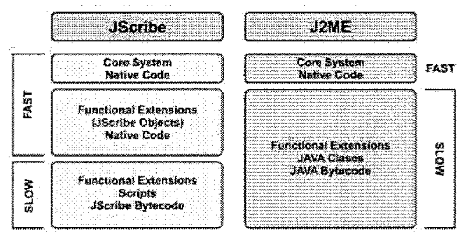
MFP features you can control with JScribe include the following:
• Any input can control any output, either locally or remotely
• The MFP can output to FTP or Email
• JScribe applications can be accessed directly from the MFP control panel
• The MFP can use serial or USB ports to allow for the integration of devices such as RFID-readers or fingerprint devices.

JScribe is deployed using two major configurations. These configurations include the following:
• Embedded JScribe - The JScribe features run directly from the MFP.
• Server-based JScribe - The JScribe features run from a server or a Windows-based PC.

Advantages over Other Embedded Platforms

Other application development platforms for MFPs use an interpreter to translate the software into a language understood by the hardware. JScribe uses the MFP's native code to perform critical functionality (such as managing print or image data). This allows JScribe applications to parse print data directly without being limited by high-level data manipulation. More importantly, it ensures significantly better overall performance compared to J2ME architectures, where almost any functionality is executed through a slow interpreter.

Services	Technology	JScribe	Other
	Language Base	Java Script/Native	Java J2ME
	Performance	FAST! Native Code	SLOW! Bytecode
	Print Data Access	Yes	No
	Native Variable Data	Yes	No
	Encryption & Print Data Compression	Yes	Encryption
	Copy, Scan, and FAX	Yes	Scan, Copy
	Control Panel Access	Yes - Platform independent	Yes - Native
	Integrated Development Environment	Yes - JScribe SDK FormMaker/...	Yes - 3rd Party Tool
	Application Range	Unrestricted	Restricted
	Platform	Embedded, Windows UNIX/LINUX, others	Embedded



SAMSUNG

JScribe Versions

JScribe 4.0

JScribe 4.0 technology provides high performance with an affordable rapid development environment. This version is currently available on the Samsung SCX 6345NJ and the ML 4551NJ. In Q3 it will be available on the Samsung SCX-6555NJ, CLS-8380NDJ, and CLS-6240FXJ MFPs.

JScribe 4.1

The next major release, JScribe 4.1, will be available after Q3 2008. This new version of JScribe includes the following new features and functions:

• **Control Panel Customization**

Control panels with graphical (color) displays can use JScribe 4.1 to not only display predefined GUI elements as JScribe 4.0 did, but also to compose new GUI elements. Compared to JScribe 4.0, 4.1 gives in-house and third-part software developers much more design freedom.

Unlike other features, which are transparent to the user, a customized control panel is immediately visible. The GUI on the control panel is the "window" to the MFP's capabilities, and it provides the human-machine interface between the user and the MFP. This interface establishes the identity of the MFP to the user and reinforces the enterprises identity.

• **Job Management**

JScribe 4.1 will provide new job management functions, which allow you to execute MFPs specific, customized jobs with higher performance and more flexibility than JScribe 4.0.

• **Extended Imaging**

JScribe 4.1 has extended the image processing engine, which not only allows image conversions between various formats, but also allows the analysis or modification of scanned or printed images. By adding watermarks or two-dimensional barcodes, increased document security is achieved for enterprise document distribution in an integrated IT environment.

Future JScribe

The evolution of JScribe is continuous process. The next generation of JScribe will support an embedded HTTP server/client, fully supporting JavaScript and allowing the Samsung MFPs to execute standardized Java applets!

Availability

JScribe 4.0 is currently available on two Samsung devices, the SCX-6465NJ and the 4545NJ. This availability is being extended to include a range of devices, which will be able to cover the needs of enterprise deployments.

This range of devices includes the following:

- Enterprise color MFPs for Letter size paper only
- Production color MFPs for Letter and Legal size paper
- Finishers such as staplers, three hole punches, brochure folders, and more.

Ready to Use Solutions

JScribe-ready MFPs can take advantage of the current catalog of JScribe solutions. These solutions have been designed to provide the functionality requested by MFP customers, and to deliver this functionality in an easy-to-install and easy-to-use package.

These JScribe solutions include the following:

- JScribe Mobile Print Solution (JMPS)
- JScribe Intelligence Server (JISS)
- JScribe Device Management (JDM)
- Embedded Applications
- Mail-Printer
- RSS-Feed Printer
- and more!

Development Tools – Overview

The power of JScribe comes from its assortment of development tools. These tools have been designed to be easily used and have been successfully implemented by sales engineers with some programming background.

Compared to the traditional firmware development process, creating a JScribe application requires much less programming knowledge and can be accomplished in a fraction of time. Once the basic JScribe skills have been acquired, creating a JScribe application is as simple as creating a Web Page.

JScribe Software Development Kit (SDK)

The SDK is an integrated software development kit, which allows you to develop fully functional JScribe applications on any Windows-based PC.

With the SDK the behavior and functionality of a Samsung MFP can be emulated by a PC. The PC will mimic a network printer and simulate the complete process of printing data. Scanning functionality can also be emulated by attaching the appropriate TWAIN compatible scanners to the PC.

The SDK editing and debugging functionality supports the complete process of creating and testing a JScribe application. Once an application is working correctly, it can be deployed to any JScribe enabled MFP, and tested directly on the hardware.

After an application has been finalized, the SDK allows you to create stand-alone deployment packages, which you can deploy throughout the enterprise.

JScribe Application Server

The JScribe Application Server (JAS) is a software emulation of JScribe, able to run on Windows PCs and

servers. A JAS license is required to enable client to server communication. By using JScribe's unique DOL® mechanism, print devices can access databases or other COM-compatible host applications through a JAS instance.

JTalk

JTalk is a license-free and royalty-free service utility, which allows you to deploy existing JScribe applications to devices, and it allows you to manage and control applications running on devices.

JScribe Developer Community JDC (Available Q3 2008)

The JDC is an online portal providing support to JScribe developers. JDC offers a wealth of JScribe technical information, application samples and other documents. The JDC download section provides access to all of the tools and documents required to bring JScribe solutions directly to the enterprise.

Samsung's B2B Solution Strategy

Providing More Standard Solutions

By providing more solutions to integrate with a growing number of JScribe compatible MFPs, Samsung will be able to cover an increasing number of customer requests by simply providing the right solution right out of the box.

Providing More Custom Solutions

The number of customers, asking for custom solution is increasing continuously. JScribe allows Samsung to provide customers with custom solutions that meet their enterprise requirements. In the past, fulfilling these customer requirements was restricted by cost and time intensive firmware modifications requiring R&D resources. JScribe allows us to significantly reduce these costs and the R&D resources needed to successfully develop prototypes and pilot applications.

API

Enterprises that want to integrate Samsung MFPs into their own solutions can use the JScribe API to access its wealth of features.

The API supports the following:

- Full JScribe functionality
- Hardware independence (production MFPs, standard MFPs, printers...).
- Open software development kit (SDK) lowering the entry barrier for any enterprise.

Since Q2, 2008, two companies have used the API to develop the following JScribe compatible solutions:

- Pitney Bowes, a leading manufacturer of folding machines and postage equipment has developed an embedded JScribe application. This application places control markers on printed documents to control the automated folding and enveloping process used for mailing letters.
- PCounter Europe, a manufacturer of printer accounting and management software is utilizing the JScribe interface to integrate Samsung MFPs into their suite of software products.

Integration

The modern customer requires more integration into their highly customized enterprise. In the B2B market it is not enough to provide the best hardware, we must also provide adaptability and compatibility. Integration capabilities, customization and "best of breed" solutions have become a requirement for success in the enterprise level B2B market.

The following projects are examples of how JScribe technology is helping Samsung to quickly address custom enterprise requirements.

Automated Document Distribution

Customer: A large government institution in the US
Number of MFPs: Over 1,000
Requirements: The enterprise needs an RFID-based user authentication feature on the MFP that is implemented using their LDAP environment. The user settings in this environment require the MFP to offer a simplified control panel interface and to allow a one-click scan distribution to either the user's home directory or to an Email address. All other network scan activities are disabled for non-authenticated users.

Stand-Alone Kiosk Copyshop

Customer: A large bank in Germany
Number of MFPs: 850
Requirements: The bank requested a stand-alone MFP (not networked), which allows a bank customer to create single-sided or double-sided copies using an extremely basic GUI. The MFP must also print a transaction invoice including a barcode to enable the customer to easily pay for their copies at the counter.

Automated Workflow Processes

Customer: One of the largest banks in Europe
Number of MFPs: 5,000
Requirements: This enterprise required an automated scan workflow where documents need to be automatically attached to a customer file.

To eliminate errors, and streamline the process for employees, the customer requested that the MFP be remotely controlled through a specialized XML protocol.

The JScribe solution displays the employees ID and contract number on the display and the only thing that the employee needs to do is to put the paper into the document feeder and select their ID and contract number. The MFP then scans all of the pages and automatically transmits them to the host, including any processing information.

Variable and Secure Form Processing

Customer: A Korean insurance company
Number of MFPs: 100 with the potential to market this solution to other insurance customers
Requirements: The enterprise must produce insurance certificates and insurance contracts for the customers. In the past they were printed in a central mass printing location and then the documents were sent by mail. A time consuming and expensive process.

A solution was developed and installed in each branch, allowing them to create the documents on demand. The content of the documents are transmitted as encrypted XML documents over the internet directly to an MFP in the branch office. The MFP prints the content of the XML file directly to a predefined form and automatically generates the documents. Two dimensional barcodes and hidden watermarks on the printed documents provide additional security against scanning and manipulating the originals.

High-Security Printing & Scanning

Customer: A European government institution
Number of MFPs: 1,400
Requirements: This public sector customer required the integration of a Smart Card authentication device and the implementation of a secure printing process. These requirements needed to be met at a low cost and in a hurry.

The output environment consisted of multiple vendors and devices. Much of the printing was performed on desktop printers and was poorly controlled.

Samsung developed a solution that allowed the institution to leverage their existing smart card security system; allowed them to control all of the MFPs from a central location; and allowed them to use audit trails that integrate with their quality system.

Questions and Answers

The following are some of the most frequently asked questions and their answers:

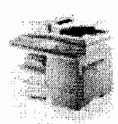
Q: Is JScribe backward compatible?
A: Yes, any application, which has been written for an earlier version of JScribe, will run correctly on new JScribe releases.
Q: Are embedded JScribe systems protected against manipulation?
A: JScribe has a built-in security system, which limits or even prevents the execution of application scripts that have not been certified by the system administrator.
Q: Can JScribe applications be protected against abuse?
A: Yes. JScribe applications can be encrypted and protected with the built-in security scheme against abuse. A protected JScribe application can only execute only on an authorized device and will refuse to run on devices which do not have the appropriate authorization.
Q: What are the minimum skills needed to create a JScribe application?
A: It is as easy to create a JScribe application as it is to compose a functional website. You also need an understanding about how an MFP works and a general knowledge about programming languages including JavaScript.
Q: How does JScribe compare to other embedded technologies on Java basis?
A: Other application platforms for printers and MFPs use an interpreter to translate the software into the language understood by the hardware. JScribe uses the native code of the device for the performance critical functionality (such as managing print or image data). This means JScribe application speed is not limited by high-level data manipulation – print data can be parsed directly, for example. But more importantly, it ensures significantly better overall performance compared to J2ME architectures, where almost any functionality is executed through an interpreter.
Q: Which impact does JScribe have on the performance of a system?
A: In principle, JScribe itself does not affect the overall performance or printing speed of a system.
However, it mostly depends on the design and functionality of the running applications whether the actual performance is affected or not. If an application i.e. has been designed for heavy network communication or needs to transfer a huge amount of data, this might impact the overall performance.

Exhibit G

SUPPORT
Owners login
MY ACCOUNT

- Ctrl P
- Printing Solutions
  - Solutions
  - Printers at a glance
  - Where to buy
- Learning
  - Why Samsung
  - Awards
  - Case studies
  - Third party reviews
- Press Room
- Support
  - FAQs
  - Online support
  - Download center

Product finder



MultiXpress 6322DN

Our thoughts on designing printers and printing solutions

As we continue to innovate and develop our printers, we'd like to share what we've learned. You'll find it all here, in our thought leadership papers.

JScribe technology

As printers have gone from only printing to complicated multi-function machines, Samsung has provided JScribe software to support all the new functions and features.

> Download PDF

Attorney Docket No.: P-74637-US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patentee(s):	KRAUTTER, Thomas	Examiner:	STEELMAN, Mary J.
Control No.:	95/001,398	Reexamination Filed:	July 16, 2010
Patent No.	6,684,789	Art Unit:	3992
Title:	METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER		

DECLARATION OF CHRISTOPH PICHT

I, Christoph Picht, declare as follows:

1. I am Director of Business Development at CCP Systems AG ("CCP"). CCP owns U.S. Patent No. 6,684,789 ("the '789 patent") that is the subject of the current reexamination.

2. I received a computer science degree (Diplom Informatiker) from Technical High School in Darmstadt, Germany, in 1981.

3. I have been working in software development since then.

4. I have been working in the electronic printing business for about 24 years.

5. I started working at CCP in April 2006.

6. I make this declaration in support of CCP's Response to the November 19, 2010 Office Action in the reexamination identified above.

7. I have read and am familiar with the '789 patent.

8. CCP has produced software called JScribe®. I am familiar with the functionality of CCP's JScribe® product.

9. JScibe® is a software product that CCP had licensed to IBM Corporation's German subsidiary. IBM's Korean subsidiary in turn sublicensed the JScribe® software to Samsung Electronics Corp., Ltd., Korea.

10. Samsung has incorporated the JScribe® software or parts of it into many of its printers and multifunctional devices.

11. Claim 1 of the '789 patent states as follows:

A method for the transformation of digital print data streams, in which

(i) an input print data stream (2) is read in,

(ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and

(iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, and

(iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (10) and are output,

characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

12. In my view, Claim 1 of the '789 patent covers the JScribe® software when used for printing. JScribe® reads in an input print stream that is analyzed by a parser. This stream is split up into graphically representable objects which are then stored in a memory in an object-oriented format. JScribe® software and any of its functionalities can be activated by scripts which can be developed easily by users, either with CCP's Software Development Kit (SDK), or with any other editor, and then be deployed to the JScribe® enabled device where they become active.

13. There are also several companies that write scripts that can be used with JScribe® software. Samsung, for example, has appointed several solution partners which use JScribe® software for integrating Samsung devices into their solutions. Among them Samsung lists MWA Intelligence, Inc., Scottsdale, AZ; SafeCom a/s, Denmark; Pcounter Europe, Denmark; Ubiquitech A/S, Denmark; Pharos Systems International, Inc., Rochester, NY; Equitrac Corporation, Plantation, FL; and X-Solutions AG, Switzerland.

14. To the extent that Samsung's printers do not include scripts assigned to objects, Samsung's printers with the JScribe® software installed have an interface that allows users to easily create scripts and to include them in the software.

15. At least one Samsung US customer (identified in Widuch Dec. Ex. E, p. 5), which according to Samsung, has "[o]ver 1000" printers, where the printers also

have scripts assigned to an object, including ones that are used for security equipment (see e.g., claims 4, 25, and 41).

16. Similarly, I believe that CCP's JScribe® software, which I understand has been incorporated into Samsung printers, has the functionality to include each of the claim elements of claims 2-16 of the '789 patent. That is, users can create scripts that interface with the JScribe® software so that they include the ability to create, or function, as follows:

a. super-objects, as claimed in Claim 2;
b. feedback messages, as claimed in Claim 3;
c. control over external devices, as claimed in Claim 4;
d. automatically to receive data, as claimed in Claim 5;
e. automatically to make data requests, as claimed in Claim 6;
f. reassign data and print out the graphic objects, as claimed in Claim 7;
g. automatically send data, as claimed in claim 8;
h. send graphic objects, as claimed in Claim 9;
i. reassign data to graphic objects and print out the graphic object, as claimed in Claim 10;
j. at least one graphically representable object is assigned to at least one script, which is executed in the case of the output of the object defined in the script, as claimed in Claim 11;
k. at least one graphically representable object is assigned at least one script, at least one case relating to the execution of the script being defined in the respective script, and occurring automatically, as claimed in Claim 12;

l. in an automatically occurring case, defined in the respective script, relating to the execution of the script, a timer, so that a case occurs automatically as a result of expiry of time, as claimed in Claim 13;

m. the timer operates cyclically, that is, it starts itself again upon expiry, as claimed in Claim 14;

n. Java Script is used as a formal language for the scripts, as claimed in Claim 15; and

o. objects stored in memory can be changed or deleted or have new objects appended, as claimed in Claim 16.

17. In addition, as explained above, systems, like various Samsung printers which include the JScribe® software, that are connected to a computer network are covered by claims 17-19.

18. For similar reasons, I believe that Samsung's printers which include JScribe® software, are covered by claim 20.

19. As explained in paragraph 15 above, at least one Samsung US customer has a print server that includes JScribe® software. There are at least half a dozen other companies that have JScribe® software loaded on their print servers. All have scripts assigned to objects that perform various functions, including, for example, having timers and the ability to reassign data. (See e.g., claims 7, 13 and 21).

20. Samsung printers, which include the JScribe® software, also are covered by claims 22-37 because they have a memory (i.e., a computer readable medium) on which the infringing method steps are stored. Scripts that users install on the printer would also be stored in memory, and these method steps would be carried out on the

printer processor when the execution command is triggered (e.g., circumstances defined in the script, for example, a timer).

21. I have reviewed the proposed new claims that CCP has submitted in this reexamination, and believe the new claims CCP has presented here cover CCP's JScribe® technology as discussed above.

22. In addition to the '789 patent, CCP obtained corresponding patents in Europe (EP 1282883 B1) and Japan (JP 3974782 B2).

I declare under penalty of perjury under the laws of the United States of America that all statements made here are based on my own knowledge and are believed to be true. I understand that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. section 1001), and may jeopardize the validity of the patent subject to reexamination.

Christoph Picht
Director of Business Development
CCP Systems AG

03.01.2011
Date

Attorney Docket No.: P-74637-US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patentee(s):	KRAUTTER, Thomas	Examiner:	STEELMAN, Mary J.
Control No.:	95/001,398	Reexamination Filed:	July 16, 2010
Patent No.	6,684,789	Art Unit:	3992
Title:	METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER		

AMENDMENT AND RESPONSE

Mail Stop Inter Partes Reexam
Attn: Central Reexamination Unit
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

This Amendment and Response is being filed by the patent owner CCP Systems AG ("CCP") in response to a Notice dated March 15, 2011 setting a 15 day response date to amend CCP's prior Response. Accordingly, this Amendment and Response is being timely filed.

Amendments to the Claims are reflected in the listing of claims below.

Remarks follow the Claims Section.

AMENDMENTS TO THE CLAIMS

Please add or amend the claims to read as follows, and cancel without prejudice or disclaimer claims indicated as cancelled:

1. (Original) A method for the transformation of digital print data streams, in which

    (i) an input print data stream (2) is read in,

    (ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and

    (iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, and

    (iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and

    (v) the objects thus transformed are combined into an output print data stream (10) and are output,

    characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

2. (Original) The method as claimed in claim 1, characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

3. (Original) The method as claimed in claim 1, characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

4. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least

one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

5. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

6. (Original) The method as claimed in claim 5, characterized in that the script (5a) automatically receiving data also requests this data automatically.

7. (Original) The method as claimed in claim 5, characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

8. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

9. (Once Amended) The method as claimed in claim 8, characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

10. (Original) The method as claimed in claim 9, characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

11. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

12. (Original) The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

13. (Original) The method as claimed in claim 12, characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

14. (Original) The method as claimed in claim 13, characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

15. (Original) The method as claimed in claim 1, characterized in that Java Script is used as a formal language for the scripts.

16. (Original) The method as claimed in claim 1, characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

17. (Original) A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates in accordance with the method as claimed claim 1.

18. (Original) The system for the transformation of digital print data streams as claimed in claim 17, the system also has an operating station with display means (8) and input means, which makes it possible for the graphically representable objects (5) stored in the memory (6) of the data processing unit in an object-oriented format, preferably also script objects (5a), to be read out via the application interface (7), to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream (10).

19. (Original) The system for the transformation of digital print data streams as claimed in claim 17, wherein the data processing unit, permits respectively stored objects, preferably also Java Script objects (5a) themselves, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations if necessary being transformed automatically into Java Script objects (5a).

20. (Original) A printer, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.

21. (Original) A printer server, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.

22. (Original) A computer-readable medium having stored thereon instructions to cause a processor to execute a method, the method comprising: (i) an input print data stream (2) is read in, (ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and (iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, (iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and (v) the objects thus transformed are combined into an output print data stream (10) and are output, characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

23. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

24. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part

objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

25. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

26. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

27. (Original) The computer-readable medium as claimed in claim 26, the method characterized in that the script (5a) automatically receiving data also requests this data automatically.

28. (Original) The computer-readable medium as claimed in claim 26, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigend by itself.

29. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

30. (Original) The computer-readable medium as claimed in claim 29, the method characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

31. (Original) The computer-readable medium as claimed in claim 30, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and forwards the graphic object (5) associated with itself to a receiver together with the data requested, received and reassigned by itself.

32. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

33. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

34. (Original) The computer-readable medium as claimed in claim 33, the method characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

35. (Original) The computer-readable medium as claimed in claim 34, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

36. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that Java Script is used as a formal language for the scripts.

37. (Original) The computer-readable medium as claimed in claim 22, the method characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

38. (Original) A computer data signal embodied in a carrier wave and representing sequences of instructions which, when executed by a processor, cause the processor to perform a method, the method comprising: (i) an input print data stream (2) is read in, (ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and (iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, (iv) the graphically representable objects (5) stored in the memory (6) in an object-

oriented format are transformed into a format for the control of an output device (9), preferably a printer, and (v) the objects thus transformed are combined into an output print data stream (10) and are output, characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

39. (Original) The computer data signal as claimed in claim 38, the method characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

40. (Original) The computer data signal as claimed in claim 38, the method characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

41. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

42. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

43. (Original) The computer data signal as claimed in claim 42, the method characterized in that the script (5a) automatically receiving data also requests this data automatically.

44. (Original) The computer data signal as claimed in claim 42, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself,

and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

45. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

46. (Original) The computer data signal as claimed in claim 45, the method characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

47. (Original) The computer data signal as claimed in claim 46, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and forwards the graphic object (5) associated with itself to a receiver together with the data requested, received and reassigned by itself.

48. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

49. (Original) The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

50. (Original) The computer data signal as claimed in claim 49, the method characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

51. (Original) The computer data signal as claimed in claim 50, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

52. (Original) The computer data signal as claimed in claim 38, the method characterized in that Java Script is used as a formal language for the scripts.

53. (Original) The computer data signal as claimed in claim 38, the method characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

54. **(New)** A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates to perform the following:

(i) read an input print data stream;

(ii) analyze the input print data steam by means of a parser for graphically representable objects and split up the input print data stream into these graphically representable objects;

(iii) store said graphically representable objects in the memory in an object-oriented format;

(iv) transform the graphically representable objects into a format for the control of a printer; and

(v) combine the transformed objects into an output print data stream, and output said combined output print data stream to said printer,

wherein said graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in cases defined in the script.

55. **(New)** The system according to claim 54, characterized in that the data processing unit is further programmed to send and receive e-mails in the cases defined in the script.

56. **(New)** The system according to claim 54, characterized in that the data processing unit is further programmed to cause original print and image data to be printed without a printer driver.

57. (**New**) The system according to claim 54, further comprising:

an operating station with display means and input means, wherein said operating system makes it possible for the graphically representable objects to be read out via the application interface, to be changed, to be deleted, or to be appended before they are output in the output print data stream.

58. (**New**) The system according to claim 57, wherein said graphically representable objects to be read out, changed, deleted, or appended via the application interface are script objects.

59. (**New**) A printer adapted for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer is adapted to transform the graphically representable objects into a format for the control of an output device, to combine the transformed objects into an output print data stream, and to output said combined output print data stream, and

wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in the cases defined in the script.

60. (**New**) The printer of claim 59, wherein said script is configured to operate as a timer, such that the script is executed automatically as a result of expiry of time.

61. (**New**) The printer of claim 59, wherein said script when executed operates to obtain information over a network, and to use the obtained information in the output print data stream.

62. (**New**) The printer according to claim 59, wherein said printer is adapted to print original print and image data without a printer driver.

63. (**New**) The printer according to claim 59, wherein said printer is further adapted to send and receive e-mails in the cases defined in the script.

64. (**New**) A printing system comprising

a printer adapted for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer is adapted to transform the graphically representable objects for the control of an output device, to combine the transformed objects into an output print data stream, and to output said combined output print data stream, and

wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in cases defined in the script; and

an operating station with display means and input means, wherein said operating station is adapted to allow manipulation of said at least one script via an application interface.

65. **(New)** The printing system according to claim 64, wherein said manipulation comprises at least one operation selected from the operations consisting of: reading out, changing, deleting, or appending said script.

66. **(New)** The printing system according to claim 64, wherein said printer is adapted to print original print and image data without a printer driver.

67. **(New)** The printing system according to claim 64, wherein said printer is further adapted to send and receive e-mails in the cases defined in the script.

68. **(New)** A printer server for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer server is adapted to transform the graphically representable objects for the control of an output device, to combine the transformed objects into an output

print data stream, and to output said combined output print data stream to the output device, and

wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in cases defined in the script.

69. (**New**) The printer server of claim 68, wherein said script is configured to operate as a timer, such that the script is executed automatically as a result of expiry of time.

70. (**New**) The printer server of claim 68, wherein said script when executed operates to obtain information over a network, and to use the obtained information in the output print data stream.

71. (**New**) The printer server of claim 68, wherein said output device is a printer.

72. (**New**) A printing system comprising

a printer server adapted for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer server is adapted to transform the graphically representable objects into a format for the control of a printer, to combine the transformed objects into an output print data stream, and to output said combined output print data stream to the printer, and

wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in cases defined in the script; and

an operating station with display means and input means, wherein said operating station is adapted to allow manipulation of said at least one script via an application interface.

73. (**New**) The printing system according to claim 72, wherein said manipulation comprises at least one operation selected from the operations consisting of: reading out, changing, deleting, or appending, said script.

74. (**New**) The printing system according to claim 72, wherein said printer server is adapted to print original print and image data on the printer without a printer driver associated with the printer.

75. (**New**) The system according to claim 72, wherein said printer server is further adapted to send and receive e-mails in the cases defined in the script.

76. (**New**) The method of claim 1, wherein said input data stream is formatted in a page description language.

77. (**New**) The method of claim 76, wherein said parser is a syntax analyzer, and wherein said analyzing by means of said parser comprises performing syntactic analysis on said input data stream.

78. (**New**) The method of claim 1, wherein storing the graphically representable objects in said object-oriented format comprising storing said graphically representable objects based on membership in hierarchically organized classes.

79. (**New**) The method of claim 1, further comprising dynamically linking a plurality of objects.

80. (**New**) The method of claim 1, wherein the objects are managed by display list management module.

81. (**New**) The method of claim 80, wherein the display list management supports one page and multi-page documents at a plurality of levels.

82. (**New**) The method of claim 81, wherein the display list management can be expanded dynamically by new objects.

REMARKS

This *inter partes* reexamination relates to US Patent No. 6,684,789 ("the '789 Patent"). The Examiner has issued an office action holding claims 1-53 invalid.

The following declarations are provided together with this response: (1) Declaration of Roland Widuch, Chief Executive Officer of CCP ("Widuch Dec."); (2) Amended Declaration of Christoph Picht, Director of Business Development at CCP ("Picht Dec."); and (3) Amended Declaration of David Birnbaum, Ph.D. ("Birnbaum Dec.").

As explained below, the claimed inventions are new, non-obvious and useful. Indeed, based on the years of commercial exploitation of the patented subject matter by both the Requester, Samsung Electronics Corp., Ltd. ("Samsung") and IBM Deutschland GmbH ("IBM"), the patented technology is certainly innovative and would not have been obvious.

Status of Claims

Claims **1-53** are subject to reexamination, and have been rejected. Claims **54-82** have been newly added in this response. The claims do not enlarge the scope of the patent or introduce new matter. Support for the new claims is provided at Appendix A.

Status of Litigation

Patentee, CCP is plaintiff in an ongoing litigation *CCP Systems AG v. Samsung Electronics Corp., Ltd. Samsung Electronics America, Inc., Samsung Networks, Inc. and IBM Corp.*, 2:09-cv-04254 (D. N.J.). CPP has accused various Samsung entities of copyright and patent infringement. The patent infringement claims have been stayed pending reexamination. CCP has also asserted a copyright infringement claim against IBM.

In the litigation, Samsung has admitted its printers have included "JScribe Core software or software adapted from JScribe Core software." (Complaint ¶ 17, attached as Appendix B). As the '789 patent explains, JScribe®, CCP's copyrighted and trademarked software, is covered by the patent in this reexamination (see '789 patent, Col. 5, lines 55-59 and Picht Dec. ¶ 11-20)

There has been no discovery in the litigation, so CCP does not know the precise extent of Samsung's infringement. But, Samsung's size is well known, and CCP believes Samsung sold at least half a million infringing printers in the United States, and many more than that on a worldwide

basis; it also improperly allowed people to download CCP's code from Samsung websites[1] (Widuch Dec. ¶¶ 5 and 6).

IBM's and Samsung's extensive, worldwide exploitation of the technology at issue – and IBM's substantial payments for the right to license the technology – together seriously undermine Samsung's claim here that the technology was allegedly old, and had been used by IBM before the filing date. Obviously, if IBM already had the technology, as Samsung now claims, IBM would never have paid to license it.

FACTUAL BACKGROUND OF THE REEXAMINATION

It is critically important to understand the historical background because it demonstrates that the Samsung's current contentions are diametrically inconsistent with its, and IBM's, pre-litigation actions.

The IBM License of JScribe® Technology and the '789 Patent

In 2003-2004, IBM evaluated CCP's technology, including the patent at issue here. (Widuch Dec. ¶ 7). After satisfying itself of the value of this technology (and inherently its innovativeness), in 2004 IBM took an exclusive license to CCP's JScribe® technology ("the IBM License"). (Widuch Dec. ¶ 7). The agreement committed IBM to pay a minimum of tens of millions of dollars in royalties over the next five years. (Widuch Dec. ¶ 7). In 2005, IBM was given the right to sub-license the technology under certain conditions. (Widuch Dec. ¶ 7).

CCP and IBM entered another agreement in mid-2004, that included a "license under any patents . . . licensable by Supplier [CCP] to make, have made, use [etc.]" which included the '789 patent. (Picht Dec ¶ 7; emphasis added).

IBM's first sub-licensee was Konica-Minolta Business Solutions, Inc. ("Konica-Minolta") which paid IBM millions of dollars in a lump-sum payment plus agreed to running royalties for the right to imbed CCP's technology into its products. (Widuch Dec. ¶ 7).

In February 2005, IBM and CCP entered into an agreement called a "Co-Marketing Logo License Agreement." (Widuch Dec. ¶ 9). That agreement set out the parties' rights vis-à-vis trademarks used in certain marketing material, and included examples of acceptable marketing materials, (an example of which is at Widuch Exhibit A). That example identifies the product as

[1] Samsung can provide the exact numbers of printers it sold that contain JScribe® technology as well as the amount of its license payments to IBM in its response, if it chooses.

"JScribe® Print Server Solution 1.0." Further, is says the product was "Developed by CCP Systems AG", and "This program is protected by copyright laws and U.S. patent 6,684,789 [the patent at issue here]." This is an admission by IBM that JScribe® software is covered by the patent at issue, and also evidence that IBM regarded the '789 patent as valid.

The Samsung Sub-License and Its Acknowledgement of Patent Validity

In about 2006, IBM's Korean subsidiary, sub-licensed the technology to Samsung. (Widuch Dec. ¶ 10). Samsung paid IBM a lump sum in the millions of dollars, and agreed to pay a running royalty for each device that included the licensed JScribe® technology. (Widuch Dec. ¶ 10). Now, of course, Samsung claims the patent is invalid.

Although JScribe® was at all times CCP's property, IBM and Samsung have at various times claimed it as their own. (discussed below pp. 21-23 and (Widuch Ex. B).

In about 2008, Samsung issued a press release entitled: "New Samsung MFP [multi function printer] with Advanced Features Offers Easy Integration for Efficient, Simple and Reliable Printing" that touted advantages of CCP's product that Samsung characterized as "*Samsung's* JScribe™". (Widuch Dec. Ex. E, p. 2; emphasis added).

In 2006, IBM awarded CCP its "Best Seller Award," a recognition of CCP's groundbreaking work on JScribe® technology. (Widuch Dec. Ex. C).

Termination of the IBM and Samsung Relationships

In January 2007, IBM sold the remainder of its printer business to Ricoh Company, Ltd. and terminated the exclusivity of the IBM License. (Widuch Dec. ¶ 13). CCP has litigation pending in Germany against IBM related to the amount of royalties that IBM was supposed to pay under the IBM License. (Widuch Dec. ¶ 13).

In 2009, CCP terminated the License with IBM, and with it any rights that Samsung may have had under that agreement. (Widuch Dec. ¶ 19). Interestingly, even after having been sued, and after bringing on this reexamination, Samsung's web site continued to promote the advantages of "JScribe technology." (Widuch Ex. G).

We discuss these indisputable facts in more detail below, but they are exceptionally strong proof of *objective* evidence that the '789 patent claims, which cover CCP's JScribe® product (Picht Dec. ¶ ¶ 11-20), would not have been obvious to those of ordinary skill. Had they been obvious, IBM and Samsung would have paid millions of dollars to license the patented technology.

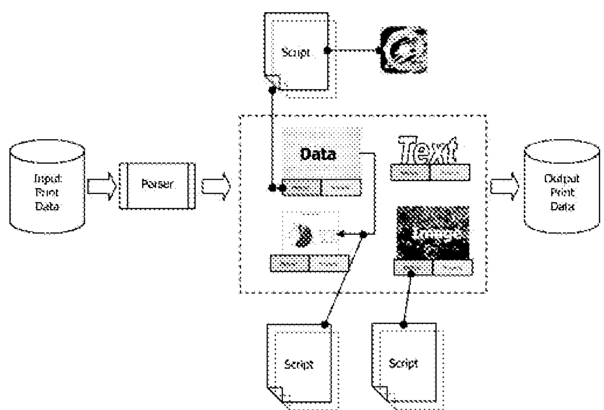
In addition to the '789 patent, CCP obtained corresponding patents in Europe (EP 1282883 B1) and Japan (JP 3974782 B2). (Picht Dec. ¶ 21).

CLAIM REJECTIONS

Introduction to the Technology

The '789 patent claims cover CCP's highly successful software product called JScribe®. (Picht Dec. ¶¶ 11-20; see also Widuch Dec. ¶¶ 7-18 and 20). Any assertion that Samsung may make that the claims do not JScribe® software, or that there has been no commercial success -- part of which is proven by Samung's action -- is frivolous.

The software, which typically resides in printers or multi-function devices, receives an input print stream and parses the print steam into objects, which are then stored in object-oriented format. (See generally, '789 patent Col. 8, lines 55-65). At least one script is assigned to an object, which is executed in the cases defined by the script. When sent for printing, the objects are transformed into a format to control a printer, and are combined into an output print data stream. This process is depicted schematically in the following figure.



By parsing the print stream into object-oriented format, as described and claimed in the '789 patent, the system attains enormous flexibility, such as allowing real-time updates (e.g., stock prices over the Web) at printing time (see '789 patent Col. 6, lines 40-41) and allows for robust two-way communication with the printer. Moreover, by using scripts, the patented software is output-device independent, allowing an organization with multiple output devices in a heterogeneous environment to control them optimally in a distributed fashion. (See '789 patent, Col. 2, line 62 – Col. 3, line 4).

Brief Summary of References

1. *IBM AS/400 Guide to Advanced Function Presentation and Print Services Facility* ("the IBM Reference"):

IBM's AS/400 is a midrange computer system. The reference describes Advanced Function Presentation ("AFP"). AFP allows graphical presentation of data produced by applications running

on the AS/400. The AS/400 receives data from an application (including text, images, etc.), prepares the data for printing (including, for example, by executing a DDS (Data Description Specification) on the data) as output print data, and sends this output data to the selected printer. (Birnbaum Dec. ¶ 8).

Requester has purported to find all claim elements in the IBM Reference, but the similarities are superficial, at best. Closer inspection of the IBM Reference reveals that essential components of claims are entirely missing, for example:

- parsing the input data stream;
- storing graphically representable objects in objected-oriented format;
- using scripts; and
- assigning at least one script to an object.

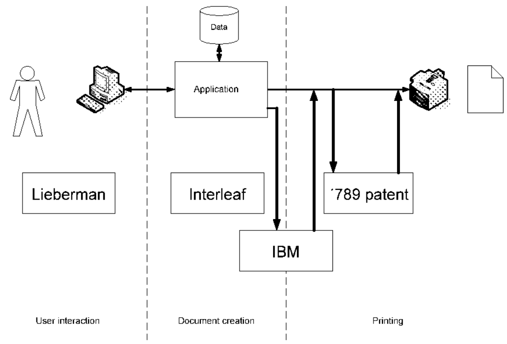
2. Interleaf Active Documents ("Interleaf") and U.S. Patent No. 5,579,519 ("Interleaf Patent") (together "Interleaf Documents")

The Interleaf Documents describe so-called "active documents." However, "active documents" are created on a computer, far upstream from the data discussed in the IBM reference (or the '789 patent) (see Figure in Birnbaum Dec ¶ 9). "Active documents" may be printed; however, the only print stream is the output of a print process, and print stream treatment is not discussed in any detail in the Interleaf documents. (Birnbaum Dec. ¶ 9). The technology in the '789 patent, by contrast, covers methodology that *prints* documents, *after* they have been created by other programs; that is, the print data stream is an <u>input</u>, which is then transformed through the patented technique into an output data stream. Transforming a print data stream is not disclosed or suggested in the Interleaf Documents.

3. "Integrating User Interface Agents with Conventional Applications", by Henry Lieberman ("Lieberman")

Lieberman describes automating user interaction by using scripts. (See Figure in Birnbaum Dec. ¶ 10). Lieberman is really far afield from the patented technology. Lieberman is not related to printing at all, and cannot be considered the same field as the '789 patent. Therefore, combining these disparate references is not something a person of ordinary skill would have done.

The following diagram shows where in the document creation and printing "food chain" these various references and the '789 patent are:



(Birnbaum Dec. ¶ 11).

As discussed in further detail below, the IBM Reference, Interleaf Documents, and Lieberman are each at different stages of the print workflow, and do not contain the claim elements. Only with improper hindsight – and complete system redesigns – would it have been "obvious" to "cherry-pick" features of each of these references to create the inventions of the '789 patent.

I. <u>Ground of Rejection No. 1:</u> (Claims 1-5, 17, 20-26, and 38-41 under 35 U.S.C. § 102(b) as anticipated by the IBM reference. See Request pp. 7-8, 10-12, and Exhibit G.)

A. IBM does not disclose a parser to analyze a print data stream

The claims recite analyzing an <u>input</u> print data stream for graphically representable objects by means of a parser. Contrary to the Request and the Office action, the IBM Reference does not disclose a parser as that term <u>is used and defined</u> in the '789 patent:

> In this case, as opposed to the use of a status machine, <u>the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions</u>, and thus powerful page description languages can also be used. (Col. 3, lines 21-26, emphasis added)[1]
>
> * * *
>
> [A] parser is used for the analysis and splitting up into the graphically representable objects . . . which is therefore <u>capable of analyzing and splitting up languages with "context-free grammars"</u>. . . (Col. 4, lines 32-38, emphasis added)[2]

A parser as properly understood according to the '789 patent, therefore, is a syntax analyzer: it is capable of analyzing the syntax of an input data stream based on a formal grammar. (Birnbaum Dec. ¶ 12).

---

[1] The term "status machine" used in the '789 patent is likely an incorrect translation from the original German and should be understood as a "state machine."

"Parser" in the '789 patent is used consistently with its well-accepted meaning in computer science. This functionality permits the parser to analyze complex page description languages (PDLs), in comparison to a state machine driven by regular expressions, which (as shown below with respect to the IBM Reference) can only reasonably handle single commands in a line by line approach. (Birnbaum Dec. ¶ 13). It should be noted that PDLs do not have "input print lines" and therefore, the statement in the IBM reference about "parsing" is not applicable to PDL input data streams, including Postscript. There is absolutely no disclosure in IBM on whether (and, if so, how) PDL inputs are "parsed" within any reasonable understanding of the term as used in the patent claims. (Birnbaum ¶ 18).

Samsung purports to find a parser at IBM reference, pp. 194-195, including:

> Specifically, the page definition defines how data is placed on a logical page layout. Input print lines are read in, optionally parsed into individual fields, and place [sic] on the page.

However, this function is capable of being performed by a state machine, and would not require a parser, as described and claimed in the '789 patent. (Birnbaum Dec. ¶ 14). This difference would have been understood by those of ordinary skill in computer science generally, and in the field of the invention in particular. Specifically, as discussed below, by referring to the data fields described at IBM Reference, pp. 14-15, it is clear that input print lines are composed of <u>structured</u> fields. Identifying and separating the individual fields from a structured field is a simpler operation than that required for parsing, and can be performed by a state machine.[2] (Birnbaum Dec. ¶ 15).

The input process described in the IBM Reference involves receiving lines in SCS (SNA [System Network Architecture] Character Stream) or IPDS (Intelligent Print Data Stream) format (defined below), and converting data fields in such lines into respective fields in AFPDS (Advanced Function Print Data Stream). (Birnbaum Dec. ¶ 16).

Regarding SCS, the IBM Reference explains:

> The SNA character string (SCS) data stream has a <u>relatively simple structure</u>, consisting of a <u>one-byte hexadecimal code followed by the data to be printed</u>. SCS, which is the standard pre-AFP print data stream, is used to control line printers and supports row and column functions. (IBM Reference, p. 14, emphasis added)

---

[2] In the same document, IBM states that "the print line is then <u>parsed (subdefined into individual fields)</u> to define the zip code field" (p. 215, emphasis added), further demonstrating the IBM Reference's loose use of the term "parsing" unrelated to syntactic analysis.

An Intelligent Printer Data Stream (IPDS) print data stream is structured according to the following format:

The IPDS Command Format

All IPDS commands are encoded in the following format:

Length	Command	Flag	CID	Data

In particular, "[t]he structured field format of IPDS allows commands to be sent to the printer in a continuous stream. Each command is self-describing." (Birnbaum Dec. ¶ 17; and Exs. B at 492 and C at 555). That is, IPDS has no grammar and no syntax across commands. (Birnbaum Dec. ¶ 17).

It is in the context of SCS' and IPDS' structured format that the cited statement in the IBM Reference, which Samsung relies on, must be understood: ("Input print lines are read in, optionally parsed into individual fields, and place [sic] on the page"). These "input print lines" are made up of structured data fields, as explained at IBM Reference, pp. 14-15. This operation of identifying and separating the individual data fields in a structured format is a different and simpler operation than parsing. (Birnbaum Dec. ¶ 18).

The result of the conversion is data in AFP data structure (AFPDS) format, which is another structured field format, shown in IBM reference Fig. 3, at p. 21, reproduced below:

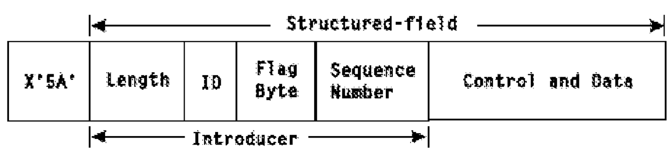
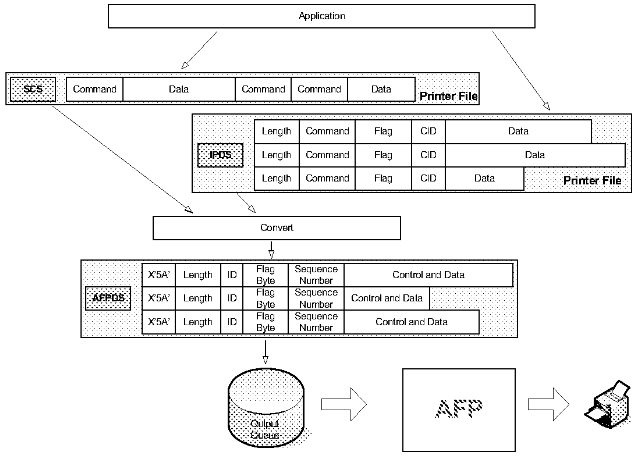


Figure 3. Structured Field

(Birnbaum Dec. ¶ 19). A field in AFPDS is identified by a leading 0X5A byte, and is followed by a number of fields that identify the length of the data, the type of data "ID", a flag byte and a sequence number, followed by the data. (Birnbaum Dec. ¶ 20). This may be understood by reference to the figure in Birnbaum Dec ¶ 16.



The critical distinction is this: **the location and content of the structured fields are predefined and do not require parsing, since their location is already defined, as is the meaning of the values in each field**. (Birnbaum Dec. ¶ 21). Thus, evaluating and processing input print lines in the IBM Reference, which is defined as structured fields, can be done by a series of if-then statements. (Id.) Such an evaluation is one implementation of a state machine, which the '789 patent expressly distinguishes from the parser used in the invention. (See '789 patent, Col. 3, lines 21-30; Birnbaum Dec. ¶ 21). Indeed, IBM says: "IPDS printers are *state machines* . . ." (Birnbaum Ex. C at 557; emphasis in original). Even though IBM AS/400 can handle Postscript documents as an input – it cannot parse them into graphically representable objects. This is because the only methodology disclosed in IBM requires conversion into structured fields. (See IBM p. 194: input print lines are "parsed into individual fields" (Birnbaum ¶ 22).

In contrast, parsing, as used in the '789 patent, is a far more complex operation; it requires scanning the input data stream character by character, identifying tokens, e.g., collections of characters that correspond to commands or data and a syntax evaluation to determine the relationships between the identified tokens. (Birnbaum Dec. ¶ 22). Often there is no rigid format for the input data stream. (Id.)

The IBM reference, in contrast, only discloses conversion from structured SCS or IPDS format to AFPDS. That function is <u>not</u> parsing under any <u>reasonable</u> interpetation of the patent claims, because it does not require syntax analysis. (Birnbaum Dec. ¶ 23). For this reason alone, the IBM reference did not anticipate the '789 patent claims.

B. IBM does not disclose storing graphically representable objects in object-oriented format

The claims recite storing graphically representable objects in object-oriented format. This too, is not disclosed in the IBM reference.

In computer science, "object-oriented programming" uses objects – i.e., data, properties and methods, together with their interactions as defined by the methods. (Birnbaum Dec. ¶ 24). Object-oriented programming techniques typically include features such as class hierarchy, data abstraction, encapsulation, modularity, polymorphism, and inheritance. (Birnbaum Dec. ¶ 25).

Indeed, the '789 patent expressly discusses object-oriented features such as object class hierarchy and inheritance:

> The individual graphic objects are stored by using their membership of specific--expediently suitably hierarchically organized—classes. . . For example, via an object of the type square, it is already known from the object hierarchy that this is a subclass of the rectangle.
>
> [I]mplicit information. . . can be derived from the object class hierarchy. . . automatically also being available to the objects of subordinate, lower-ranking classes by way of inheritance. . . (Col. 3, line 38 – Col. 4, line 8, emphasis added)

Therefore, object-oriented format as used in the '789 patent, and as generally understood in computer programming, requires that objects have the ability to be organized, for example, in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend. (Birnbaum Dec. ¶ 26).

Some advantages of using object class hierarchy are described in detail in the '789 patent:

> The object-oriented . . . format . . . makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended. . . by assigning the methods required for this to the respective objects in accordance with their class hierarchy. (Col. 7, lines 24-30, emphasis added)
>
> * * *
>
> [F]eedback messages referring to the output print data stream output are read in and are analyzed for error messages which indicate that the output device. . . has recognized a transformed graphic object in the output print data stream which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity. . . (Col. 4, lines 40 – 46).

If, for example, the printer is not capable of recognizing and outputting a bar-code object "then the bar code is simply split up into objects of the next lower hierarchy and a further try is made with these objects." This is continued, if necessary, until the printing is successful. "The object-oriented data structure with its object hierarchy, chosen for the intermediate format, also proves to be particularly suitable for this procedure." (Col. 4, lines 50-61).

Samsung purports to find "object-oriented architecture" at IBM Reference, pp. 21, 23, and Fig. 5. However, the IBM Reference uses the term "object" in a wide variety of ways, as exemplified by the thirteen definitions of the word in IBM's IBM Dictionary of Computing, 10th Ed., 1993; copy at Appendix C. The IBM dictionary separately defines an "object" in the context of object-oriented design as distinguished from its use in connection with the AS/400:

> object. . . (10) In object-oriented design or programming, an abstraction consisting of data and the operations associated with that data. . . (11) In the AS/400 system, a named storage space that consists of a set of characteristics that define itself and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed; for example, programs, files, libraries, and folders. (See Appendix C).

The IBM Reference uses the term "object" to refer to data structures in which various document elements (e.g., image, form, text, font, bar code, and graphic) are represented. For a number of reasons -- including lack of: class hierarchy, assignment of methods to objects and inheritance -- "object," as used in the IBM Reference, does not have the same meaning as "object" in object-oriented programming, and as that term is used in the '789 patent. (Birnbaum Dec. ¶ 27). Indeed, the structured field format of AFPDS may be suitable for storing such data elements, but <u>it is not suitable for storing objects in object-oriented format</u>. (Id.)

So, for example, page 23 of the IBM Reference, purports to describe as an "object" an image of an airplane. However, that so-called "object" is not handled as a member of a class to which dedicated methods are assigned – and therefore cannot be considered an object in the object-oriented programming sense. (Birnbaum Dec. ¶ 28).

Moreover, the IBM Reference uses Mixed Object: Document Content Architecture-Presentation (MO:DCA-P) (IBM, p. 21).

> IBM has defined a single object-oriented data stream—Mixed Object Document Content Architecture (MO:DCA). (<u>An object is a collection of data that can be treated as a unit</u>.) (Emphasis added).

See, Birnbaum Dec. Exs. B 495 and C at 558.

The IBM Reference discloses simple data structures whose data are treated <u>as a unit</u> – and nothing more. (Birnbaum Dec. ¶ 29). By contrast, an object-oriented format, in the sense of the '789 patent and elsewhere, requires that objects be capable of being organized in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend. (Birnbaum Dec. ¶ 30).

When all the evidence is carefully considered, the IBM Reference does not disclose graphically represented objects stored in object-oriented format, a limitation in all claims of the '789 patent.

C. The IBM Reference does not disclose use of a script

The claims recite that a stored script is assigned to a graphically representable object. The Requester has pointed to the IBM reference and claimed that the Data Description Specification ("DDS") is a script. However, this is based on a fundamental misunderstanding of the meaning and importance of a script (and of how DDS operates).

A script – as distinguished from other types of programming languages is a program or series of commands that is interpreted and runs in real time rather than compiled and then executed. (Birnbaum Dec. ¶ 31). According to IBM's definition, scripts are defined as: "[a] high-level programming, or command, language that is interpreted (translated on the fly) rather than compiled ahead of time." And, "[s]cripts are interpreted as they are run." (Birnbaum Dec. ¶ 32). In contrast, programs written in standard program languages, such as FORTRAN, C++, etc., must typically be compiled into object code before being executed. (Birnbaum Dec. ¶ 33). This distinction is important to attain the benefits of the claimed inventions, in particular: flexibility and portability. (Birnbaum Dec. ¶ 34).

One feature of scripts is that they may be freely modified, including at run-time, which is not true for compiled code. For example, in one embodiment of the invention, the data processing unit permits stored objects, including particularly script objects, to be read out graphically, to be changed, to be deleted or to be appended. See, e.g., claims 18, 19. This can only be done using a script, not compiled object code. (Birnbaum Dec. ¶ 35).

Further, object code is limited because it is platform dependent. (i.e., a particular processor architecture). In contrast, any processor capable of interpreting a script can run the script. This renders scripts portable, i.e., platform independent. (Birnbaum Dec. ¶ 36).

The Request purports to find a script in IBM's use of Data Description Specification (DDS). However, the IBM Reference does not use the word "script", nor does it refer to DDS as "scripts"; on the contrary, DDS must be compiled before being used:

> The DDS will be **compiled** before the application program runs. The application program never looks at the DDS file and member, only at the **compiled** results. . . You can update the DDS; however, **you must then re-compile it**.

Birnbaum Dec Ex. B at 61 and Ex. C at 61 (emphasis added).

This means the DDS is compiled some time before being used for printing, and can only be executed as object code after being compiled, and unlike scripts must be re-compiled if modified. (Birnbaum Dec. ¶ 37).

This feature of DDS is a critical distinction from a script, in which interpretation and execution of the script are typically simultaneously triggered, in the case of the '789 patent, for example, by the incoming print data. (Birnbaum Dec. ¶ 38).

Therefore, contrary to Samsung's arguments, the DDS described in the IBM Reference are not scripts as claimed in the '789 patent, or under any sensible interpretion of the word, nor are they even referred to in the IBM Reference as such.

D. The IBM Reference does not disclose a script assigned to an object

All claims require that at least one script be assigned to an object (stored in an object-oriented format). As discussed above, the "elements" in the IBM Reference (images, text, etc.) are not objects stored in object-oriented format. Even if an "element" were an object, and DDS were a script (which they are not), the DDS is not assigned to an element, but rather is applied to the document as a whole. (Birnbaum Dec. ¶ 39).

The '789 patent explains by assigning a script to a particular object, various benefits and advantages may be obtained, including, for example (see e.g., col. 5, lines 27-31 and col. 6, lines 38-51):

- automatically sending, reuesting or receiving data, e.g., data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.
- reassigning data received to the graphic object associated with it.
- forwarding the graphic object associated with itself to a receiver together with the data requested, received and reassigned by itself.

The AS/400 machine described in the IBM Reference cannot perform these functions, and would have to be entirely reconfigured, and the code totally rewritten to perform these operations. (Birnbaum Dec. ¶¶ 40-41).

Because independent claims 1, 22, and 38 are allowable over the IBM Reference, dependent claims 2-5, 17, 20, 21, 23-26, and 39-41, which depend directly or indirectly thereon, are likewise allowable.

II. Ground of Rejection No. 2: (Claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, and 48-49 under 35 U.S.C. § 103(a) as obvious over the IBM reference in view of Interleaf. See Request pp. 5, 7-8, and Exhibit H)

A. There is no Prima Facie Case of Obviousness

A prima facie case of obviousness requires a showing that the invention as claimed would have been obvious at the time of the invention to one having ordinary skill in the art. It would not have been obvious in 2000 to combine the IBM and Interleaf references, and in any event, such combination would not have resulted in any of the claimed inventions.

Although, the Examiner "can take account of the inferences and creative steps that a person of ordinary skill in the art would employ," *KSR International Co. v. Teleflex Inc.*, 127 S.Ct. 1727, 1741 (2007), it is critically important when undertaking this analysis to keep firmly in mind that "the person of ordinary skill in the art is an objective legal construct presumed to think along conventional lines without undertaking to innovate, whether by systematic research or by extraordinary insights." Life Technologies, Inc. v. Clontech Laboratories, Inc., 224 F.3d 1320, 1325 (Fed. Cir. 2000); Std. Oil Co. v. Am. Cyanamid Co., 774 F.2d 448, 454 (Fed. Cir. 1985) ("A person of ordinary skill in the art is also presumed to be one who thinks along the line of conventional wisdom in the art and is not one who undertakes to innovate, whether by patient, and often expensive, systematic research or by extraordinary insights, it makes no difference which." (Emphasis added)). This person does not look for exceptionally creative solutions, and does not consider how to completely redesign existing software to arrive at novel, and admittedly extremely useful products that were previously unknown.

Indeed, as explained in MPEP (§ 2142):

> To reach a proper determination under 35 U.S.C. 103, the examiner must step backward in time and into the shoes worn by the hypothetical "person of ordinary skill in the art" when the invention was unknown and just before it was made. In view of all factual information, the examiner must then make a determination whether the claimed invention "as a whole" would have been obvious at that time to that person.

Accord, *KSR* at1742 ("A factfinder should be aware, of course, of the distortion caused by hindsight bias and must be cautious of arguments reliant upon ex post reasoning. See *Graham* [*v. John Deere Co. of Kansas City*], 383 U.S. [1], 36 [1966] [] (warning against a 'temptation to read into the prior art the teachings of the invention in issue' and instructing courts to 'guard against slipping into the use of hindsight')).

1. Defining the Field of the Invention and One of Ordinary Skill

First, the field of the invention and the person of ordinary skill in the art must be defined. That determination is based on what is claimed. As the MPEP explains (§ 904.01(c)) (emphasis added):

> It depends upon the necessary essential function or utility of the subject matter covered by the claims, and not upon what it is called by the applicant.

All claims at issue expressly cover transformation of print data, in particular parsing input print data streams. The field of the invention is not as Samsung suggests, document creation (Interleaf) or user interfaces (Lieberman), because the structure and function of those references are totally unrelated to the subject matter and claims of the '789 patent, and therefore are plainly non-analogous arts. See MPEP § 2141.01(a) II.

Obviously, the relevant field cannot be the entire field of computer science, which essentially is what Samsung is arguing.

The field of the invention of the '789 patent is, printing software generally, and in particular, interpretation or processing of print data streams at a printer or print server. (Birnbaum Dec. ¶ 53). A person of ordinary skill working in the relevant field in May 2000 (the earliest priority date) or as late as May 2001 (the PCT filing date), would have been a person with a computer science degree (typically, a bachelor's degree), and approximately 2-4 years of experience. (Birnbaum Dec. ¶ 55). This experience may typically have been acquired working at a company making printers or printer software, such as Xerox, Hewlett-Packard, etc. (Birnbaum Dec. ¶ 56).

2. The IBM and Interleaf References are in Entirely Different Fields

These principles mean that a person of ordinary skill would not have combined the Interleaf references with the IBM reference.

The IBM reference describes how data from an application running on the AS/400 may be formatted (e.g., using a DDS) and sent to a printer. It describes providing print capability to networks controlled by a central server, like the AS/400. (Birnbaum Dec. ¶ 58).

However, Interleaf deals with "document creation" and manipulation software (Interleaf, p. 75).[3] They are completely unrelated to the invention. (Birnbaum Dec. ¶ 59). The Examiner points

---

[3] The Interleaf reference is discussed in detail in connection with Ground of Rejection No. 5.

to Interleaf's brief discussion of printing an Interleaf document. (Office action, pp. 18, 27). However, this merely refers to a print function to generate a print stream, presumably in a known print data stream, such as a page definition language. It is completely irrelevant to Interleaf what happens to the print data downstream. That is, Interleaf does not involve interpretation of the print data streams at a printer or print server. (Birnbaum Dec. ¶ 60).

Further, people involved in document creation software and those working in printer software or programming would typically have been working in different divisions or departments, or more likely in different companies altogether. (Birnbaum Dec. ¶ 57). One of ordinary skill in the field of printing would not have looked to Interleaf , and its disclosure of document creation, in order to solve problems involving print data streams. (Birnbaum Dec. ¶ 61).

Moreover, the USPTO ascribed the '789 patent and the Interleaf Patent to entirely different fields: the U.S. Class/Subclass assigned to the '789 and Interleaf patents (and their fields of search) highlight their differences (See Appendix D). See MPEP § 2141.01(a) II ("While Patent Office classification of references and the cross-references in the official search notes of the class definitions are some evidence of 'nonanalogy' or 'analogy' respectively, the court has found 'the similarities and differences in structure and function of the inventions to carry far greater weight.'"). Here, there is no overlap between the classifications – and, as explained above, no overlap between either the structure or the function of the prior art references and the '789 patent.

Additionally, Samsung has proposed, and the Examiner has incorrectly adopted, two different fields of the inventions depending on which prior art reference Samsung was trying to match up; Samsung's two fields of the invention are inconsistently defined as follows:

(1) "object oriented document publishing systems with scripting functionality,"[4] and

(2) "scripting interfaces for object systems".[5]

Obviously, the field of the invention is defined by the patented invention itself, not the prior art an infringer is seeking to combine, as Samsung, has done here. That Samsung has proposed two different fields of the invention itself demonstrates Samsung's conclusions are baseless.

Moreover, (1) is not a definition of a field of art that existed *prior* to the patent's priority date, but rather a hindsight rephrasing of keywords from the '789 patent, which does not reflect the realities of the programming world. See MPEP § 2142 (cited above). Number (2) is equally odd.

[4] See Exhibit H (p. 11; to combine IBM and Interleaf, and adopted by the Examiner on pp. 11, 17 and 41).

[5] See Exhibit J (p. 21; to combine Interleaf and Lieberman, and adopted by the Examiner on pp. 26, 33 and 39).

The patented technology is not a generic "object system" – even assuming that phraseology has ever existed anywhere but Samsung's papers. CCP knows of no field that existed in 2000 described as set forth in (1) or (2). (Birnbaum Dec. ¶ 54).

The USPTO's classifications cited above for the Interleaf Patent and the '789 Patent do not include either of these supposed fields.

The obvious conclusion is that Samsung's "fields" are made up for the purpose of this reexamination and are improper. What therefore becomes plainly apparent is Samsung's references have been shoehorned into Samsung's artificial "fields" for the sole purpose of trying to show invalidity in this reexamination. In a related instance, then Chief Judge Markey of the Federal Circuit held:

> Appeals in patent cases should not be mere games played with pieces of paper called references and the patent in suit. Lawsuits arise out of the affairs of people, real people facing real problems. So here, the technical problem out of which grew the present litigation was real. It was solved by inventor Cardeiro.

Rosemount, Inc v. Beckman Instruments, Inc., 727 F.2d 1540, 1544 (Fed. Cir. 1984).

Samsung is indeed "play[ing] with pieces of paper called references" that do not relate to the problem that CCP solved. Samsung is looking for keywords in documents that do not perform the claimed functions, slapping these references together, and concluding that someone else would have invented the technology at issue earlier. The fact is no one did, and the party that wrote Samsung's Key reference, IBM, paid handsomely for the right to use the patented technology.

When properly understood, the Interleaf Reference (and patent) are in an entirely different fields of technology than the '789 Patent. One of ordinary skill, as correctly defined, would not have referred to the Interleaf Documents or combined them with the IBM Reference to arrive at the inventions of the '789 Patent. (Birnbaum Dec. ¶ 61).

3. There Would Have Been no Motivation to Combine IBM and Interleaf

Adopting Samsung's request, the Examiner purported to explain the motivation for combining IBM with Interleaf as follows:

> Combining the print process of IBM that parses and transforms a print stream into an object-oriented format, and the print-scripting process of Interleaf are combining well known printing techniques to yield predictable results.

Office action, pp. 11-12, citations omitted.

As discussed below, this is pure hindsight reconstruction, and is not based in technological realities at the time of the invention. Moreover, there is no "print-scripting process of Interleaf." (Birnbaum Dec. ¶ 63). Neither of the Interleaf Documents discloses that functionality. (Id.). This, therefore, is a flawed argument. Conceivably, after extensive redesign, the Interleaf and IBM references might have been linked, but not combined. (Birnbaum Dec. ¶ 64). Let us assume, *arguendo*, that the Interleaf system were operating on a PC connected to an AS/400 midrange computer. A user *might* be able to operate the Interleaf system on the PC, and then send a document to the AS/400 to print. The AS/400 would then process the data, for example, by formatting it using a DDS. The result, however, would not be the inventions claimed in the '789 Patent because any object-oriented formatting, including any scripts, in the Interleaf "active document" would be lost prior to generating the print data stream that would be transmitted to the AS/400. Thus, even in a "linked" system, the AS/400 print data stream could not have either: (a) object-oriented formatting, or (b) scripts disclosed in the Interleaf references. (Birnbaum Dec. ¶ 64).

Moreover, in order to combine the features of Interleaf and IBM, one would have had to redesign the AS/400 AFP by taking the entire object-oriented structure, together with the scripts, described in Interleaf, and import them onto the AS/400, where the printing facilities reside. (Birnbaum Dec. ¶ 65). There is absolutely no suggestion to do this – nor is there any reason to do this. It would result in an entirely reconstructed system. See, e.g., MPEP § 2143.01(VI) ("If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references <u>are not sufficient to render the claims *prima facie* obvious</u>." Emphasis added)

Samsung's purported motivation to combine is that "IBM provides external printing and publishing functionality. . . IBM also teaches the desire to control outside printing and publishing functionality. . ." (Office action, pp. 11-12). The "external" functionality of Interleaf refers to printing, whereas the "external" functionality of IBM refers to staplers and binders, neither of which requires scripting functionality or object oriented code. (Birnbaum Dec. ¶ 66). Again, common words (without a common meaning) cannot, contrary to Samsung's assertions, be the basis for a motivation to combine the references.

As the Supreme Court held in *KSR*: "'Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness'"; accord, MPEP § 2142 ("rejections on obviousness cannot be sustained with mere conclusory statements.")

As discussed above, the AS/400 operates on data using a compiled DDS, not a script. It would not have been obvious – even in light of Interleaf – to modify the AS/400 to use scripts. (Birnbaum Dec. ¶ 67). The AS/400 is a centralized platform, and therefore, would not have benefitted from the platform-independence of scripts. (Birnbaum Dec. ¶ 68). On the contrary, scripts may often be slower to execute than object code, because they must be interpreted in real time. (Birnbaum Dec. ¶ 69). Samsung has not shown that one or ordinary skill would have been motivated to replace the compiled DDS with a script to be interpreted at run time.

Samsung's conclusion is that a "person of ordinary skill in the art would be motivated to 'take full advantage' of the printing and publishing capabilities of the IBM system by combining it with the Interleaf system." (Office action, pp. 11-12). This is not a motivation; it is a play on words, devoid of technological substance, and assumes the conclusion. There are many, many ways to "take full advantage" of IBM's capabilities that have nothing to do with the inventive technology (e.g., using a faster printer, a color printer, larger or faster memory, etc.).

4. The IBM and Interleaf References Even if Combined Would Not Have Resulted in the Inventions of the '789 Patent

Finally, even if the Interleaf system operating on a PC were linked to an AS/400 midrange computer, it would not have resulted in the present invention. In Samsung's combination, the print output of such a PC would immediately have been converted to the structured AFP format in order to be manipulated by the AS/400 (Birnbaum Dec. ¶ 71). The IBM reference states:

> Network applications are generating document data streams, such as Postscript, and image formats, such as GIF and TIFF. <u>Data in these formats can be converted to AFP and then stored or printed from the AS/400</u>. (IBM, p. 8, emphasis added).
>
> PostScript is a data stream generated by many PC and network station applications. . . If that PostScript data stream is being spooled to the AS/400, <u>the data stream must be converted to AFP data stream in order to print on an IPDS printer</u>. (IBM, p. 16, emphasis added).

Using the Interleaf software on a PC client connected to an IBM AS/400 would not cause the AS/400 to store graphically representable objects in object-oriented format, or assign a script to an object. Adding Interleaf to the "front end" of the IBM reference – simply results in printing documents as disclosed in the IBM Reference. Nothing in the print stream is changed: there still are no objects, no scripts assigned to objects and no parser. (Birnbaum Dec. ¶ 72).

Because the IBM Reference and Interleaf do not individually, or in combination, disclose the elements of independent claims they do not disclosure, or render obvious any of the dependent claims either.

B. Objective Evidence of Non-Obviousness Overcomes Any Purported Showing to the Contrary

Even if there were a *prima facie* case of obviousness, it is far outweighed by objective evidence of non-obviousness presented in the Widuch and Picht Declarations. "Once the applicant has presented rebuttal evidence, Office personnel should reconsider any initial obviousness determination in view of the entire record." MPEP § 2141(V). This is the stage where we are now.

Secondary considerations of non-obviousness must be considered. *Transocean Offshore Deepwater Drilling, Inc. v. Maersk Contractors USA, Inc.*, 617 F.3d 1296, 1305 (Fed. Cir. 2010); see MPEP § 716.01(a) ("Affidavits or declarations. . . containing evidence of criticality or unexpected results, commercial success, long-felt but unsolved needs, failure of others, skepticism of experts, etc., must be considered by the examiner in determining the issue of obviousness of claims for patentability. . ."); § 2154 ("Evidence pertaining to secondary considerations must be taken into account whenever present[.]") (Emphasis added throughout).

> Evidence of secondary considerations may often be the most probative and cogent evidence in the record. It may often establish that an invention appearing to have been obvious in light of the prior art was not. It is to be considered as part of all the evidence, not just when the decisionmaker remains in doubt after reviewing the art.

Stratoflex, Inc. v. Aeroquip Corp., 713 F.2d 1530, 1538 (Fed. Cir. 1983) (emphasis added, internal quotation marks omitted).[6]

The current obviousness determination cannot survive where the owner of the allegedly invalidating prior art spent millions of dollars on the patented technology and where the requestor also paid millions of dollars in licensing fees and praised the patented product. *WMS Gaming, Inc.*

---

[6] Indeed, the objective evidence of non-obviousness may overcome a prima facie case of obviousness. *Gambro Lundia AB v. Baxter Healthcare Corp.*, 110 F.3d 1573, 1580 (Fed. Cir. 1997) (reversing finding of invalidity where district court failed to consider objective evidence of non-obviousness, including commercial success and failure of others); *Ruiz* at 667; *Hughes Tool Co. v. Dresser Industries, Inc.*, 816 F.2d 1549 (Fed. Cir. 1987) (affirming validity based on objective evidence of non-obviousness); *Simmons Fastener Corp. v. Ill. Tool Works, Inc.*, 739 F.2d 1573, 1575, (Fed. Cir. 1984) (reversing holding of invalidity and finding that the objective evidence overcame the lower court's decision on obviousness, holding that "[o]nly after all evidence of nonobviousness has been considered can a conclusion on obviousness be reached.").

v. Int'l Gaming Tech., Inc., 184 F.3d 1339, 1359 (Fed. Cir. 1999); *see also, Ruiz v. A.B. Chance Co.*, 234 F.3d 654 (Fed. Cir. 2000).

Because this objective evidence arises out of economic and business realities, and is not subject to the vagaries of litigation inspired technical arguments, it often is the most probative and cogent evidence in the record.

Samsung's use of the patented technology certainly demonstrates commercial success; Samsung's and IBM's comments show praise in the industry; and Samsung's illegal use following termination of the IBM License and its appropriation of the name "JScribe" also shows copying, all important objective of non-obviousness that easily overcome any *prima facie* claim of obviousness. This evidence <u>must</u> be considered in any obviousness analysis, *Transocean* at 1305 (Fed. Cir. 2010); see MPEP §§ 716.01(a), 2154, and reverses the Examiner's initial conclusion.

1. Commercial Success and Licensing Supports Validity of the '789 Patent

As discussed above, in 2004 IBM's German subsidiary entered into the IBM License for CCP's JScribe® technology, covered by the '789 patent, and committed IBM to pay tens of millions of dollars. (Widuch Dec. ¶ 7).

Konica-Minolta also paid IBM millions of dollars as a lump sum for the technology and agreed to pay a running royalty to IBM, of which CCP received 50%. (Widuch Dec. ¶ 8).

Samsung paid IBM several million as a lump sum and was supposed to pay IBM a running royalty for each device that included the patented JScribe® technology. (Widuch Dec. ¶ 10).

None of these facts can be seriously disputed.

2. Praise in the Industry Supports Validity of the '789 Patent

In *Gambro Lundia AB v. Baxter Healthcare Corp.*, 110 F.3d 1573, 1579 (Fed. Cir. 1997), the infringer (Baxter),

> touted the advantages of [the patented feature] AUTO-ADJUST, as it termed automatic recalibration during dialysis, in the advertising for the allegedly infringing Baxter SPS 550 machines. Baxter's recognition of the importance of this advance is relevant to a determination of nonobviousness.

Accord Eli Lilly & Co. v. Zenith Goldline Pharmaceuticals, Inc., 471 F.3d 1369 (Fed. Cir. 2006) (pointing to the record showing a number of awards as indicators of industry acclaim). See also, *In re Tiffin*, 443 F.2d 394, 400 (CCPA 1971):

> Recognition by the trade is the best and most persuasive evidence that can be offered. The tribute of those engaged in the industries affected,

> especially when the tribute is evidenced by the payment of substantial royalities [sic], is by far the most persuasive and unimpeachable evidence that can be offered to support the asserted validity of patent claims in litigation.

In this regard, IBM and Samsung issued a joint press release that stated: "'JScribe', the printing optimized middleware of IBM will be launched in the Samsung digital printing devices such as printers and multifunction products." (Emphasis added) (Widuch Dec. Ex. B). It is unclear which of the two companies (or both) was responsible for drafting this document, but JScribe® software has always been a CCP product. Nevertheless, IBM and/or Samsung took credit for the innovative technology.

In 2006, IBM awarded CCP its "Best Seller Award," a recognition of CCP's groundbreaking work on JScribe® technology. (Widuch Dec. Ex. C).

In June 2007, CCP's JScribe® product won the European Commission China Information and Communications Technologies Innovator Award (ChinICT). (Widuch Dec. Ex. D).

In about 2008, Samsung issued another press release entitled: "New Samsung MFP with Advanced Features Offers Easy Integration for Efficient, Simple and Reliable Printing" that discussed the many advantages of the CCP product that Samsung characterized as "Samsung's JScribe™". Samsung obviously does not own the trademark to JScribe. But, it is interesting that Samsung appropriated this important technology as its own, technology that it now belittles. (Widuch Dec. Ex. E, p. 2).

In that press release, Samsung also proclaimed the benefits of CCP's JScribe® product (See Widuch Dec. Ex. E, p. 2):

- "open platform technology means these new features can be added quickly and easily in response to the individual needs of the customer."
- "Corporate users can maximize their printing investment by customizing the device to their specific requirements."

In about 2008-09, Samsung gave the patented JScribe® technology effusive praise in a "White Paper" only some of which is listed below (See Widuch Dec. Ex. F for all statements):

- "highly flexible and portable to almost any microprocessor-based system" (p. 1);
- "A simple control script can serve as the device driver and can control a broad range of devices including: fingerprint readers, Card readers, RFID printers, RFID readers, label printers, Slave printers." (p. 2).

- "Print job filter objects allow JScribe applications to recognize incoming print data and, if required, modify the print sequences before the print data reaches the MFP's RIP [Raster Image Processor]." (p. 2).
- "In the past, modifying the printer code was either impossible or extremely expensive." (p. 3).
- JScribe uses the MFP's native code to perform critical functionality (such as managing print or image data). This allows JScribe applications to parse print data directly without being limited by high-level data manipulation." (p. 3).

Samsung identified seven lines of printers that include CCP's JScribe® product and a number of customers that used thousands of Samsung printers with JScribe® technology installed, including "a large government institution in the US". (pp. 4-6). In CCP's view, all the deals cited in the Samsung White Paper (Widuch Dec. Ex. F) could only have been done because of the JScribe® technology in Samsung's devices. (Widuch Dec. ¶ 17).

3. Copying by Others Supports Validity of the '789 Patent

In 2009, CCP terminated the License with IBM, and with it any rights that Samsung may have had under that agreement. (Widuch Dec. ¶ 19).

Even after having been sued, and after bringing on this reexamination, Samsung's web site continued to promote the advantages of "JScribe technology": "As printers have gone from only printing to complicated multi-function machines, Samsung has provided JScribe software to support all the new functions and features." (Widuch Ex. G).

* * *

To summarize: at least two critical and undeniable facts stand out. First, IBM, on whose technology Samsung relies in this reexamination, spent million dollars to acquire rights to the patented technology and paid on-going royalties to CCP for the privilege of licensing and sub-licensing that technology. Second, Samsung, the requester here, paid IBM, and IBM paid the patent owner, CCP, millions of dollars for the rights to license and use the patented technology. (Widuch Dec. ¶¶ 7, 10).

Despite all this contrary evidence, Samsung now says that IBM's AS/400 software is identical to the '789 patent (i.e., IBM anticipated the claimed subject matter) and rendered obvious the innovations claimed in the '789 patent. But the truth of the matter is that in 2004, four years

after the patent's priority date, and five or more years after publication of the IBM Reference, IBM paid handsomely to license CCP's software covered by the patent at issue. (Widuch Dec. ¶ 7).

As a matter of common sense, if the innovations of the '789 Patent were so obvious (as Samsung now claims), and IBM already had identical or similar software, neither Samsung nor IBM would have paid anything to license CCP's software.

III. Ground of Rejection No. 3: (Claims 1-14, 16-18, 20-35, 37-51 and 53 under 35 U.S.C. § 103(a) as obvious over the IBM reference in view of Interleaf and the Interleaf Patent. See Request pp. 10-12, and Exhibit I).

The Interleaf Patent does not add relevant subject matter to the Interleaf reference. The Interleaf Patent, too, deals with document creation and editing. (Birnbaum Dec. ¶ 80). At most, the software described in the Interleaf Patent may generate a print stream, but it does not interpret or transform a print stream, as claimed in the '789 patent. Therefore, the same reasons, presented above, why the references would not have been combined, and in any event, would not have resulted in the inventions of the '789 patent if combined, are applicable here. The objective evidence of non-obviousness is likewise applicable.

IV. Ground of Rejection No. 4: (Claims 15, 19, 36, and 52 under 35 U.S.C. § 103(a) as obvious over the IBM reference in view of Interleaf and further in view of the Interleaf Patent and Lieberman. See Request pp. 10-13, and Exhibit I).

Lieberman is about automating user interaction by using scripts. The paper reports "experiments in developing agent software that works with existing unmodified commercial applications and agents that work across multiple applications."

Neither Samsung's request for reexamination nor the Office action explains how Lieberman relates to the field of the invention, interpretation of print data streams at a printer or print server. Lieberman has nothing to do with printing, print streams, printers, or print servers. There would have been no reason for one of ordinary skill in the art to refer to the Lieberman reference, or to use its teachings to solve the problems addressed in the '789 patent. (Birnbaum Dec. ¶ 96).

Accordingly, the obviousness determination based on any combination involving the Lieberman reference is simply wrong – and is the result of improper hindsight analysis.

V. <u>Ground of Rejection No. 5:</u> (Claims 1-2, 4-7, 11-12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 under 35 U.S.C. § 102(b) as anticipated by Interleaf. See Request pp. 10-14, and Exhibit I).

Because IBM and Interleaf combined do not have all the claim elements, *a fortiori*, Interleaf alone cannot have all the necessary elements. (See Ground No. 2).

Interleaf is a "document creation component" (Interleaf, p. 75). It builds an enhanced user interface (UI). ("Use of this natural UI makes this produce easy to learn and hence extremely popular." Interleaf, p. 76). As explained above, the '789 Patent addresses an entirely different part of the workflow, namely, the printing stage. Consequently, there are at least four fundamental differences between Interleaf and the '789 Patent. Interleaf does <u>not</u>:

- disclose a method for the transformation of print data streams;
- read in an input print stream;
- parse an input print data stream; or
- produce an output print data stream based on a transformed input data stream.

(Birnbaum Dec. ¶ 99).

A. Interleaf Does Not Disclose Transformation of Print Data Streams

Claims 1, 22, and 38 recite a method, computer-readable medium, and computer signal for the transformation of digital print data streams. The Examiner purported to find this claim element at Interleaf, p. 76: "I6 also has font and other style tokens that can be placed in the middle of a component text stream to change style without requiring an inline component." In addition, the Examiner has further pointed to pp. 81-84, regarding changing objects in a document, e.g., changing color or size or position of the object. But these refer to editing a working document in the user interface, not printing it; that is, these changes are performed to edit or create the document -- not to print it. (Birnbaum Dec. ¶ 100).

These portions of Interleaf merely describe users creating or editing a document, not what happens at print time. Therefore, they cannot describe transformation of a print data stream. (Birnbaum Dec. ¶ 100).

B. Interleaf Does Not Read in an Input Print Stream

The claims recite that an input print data stream is read in. The Examiner purported to find this element at Interleaf p. 84 ("If a document contains active objects within the document file stream. . . these become activated when the document is <u>opened</u> programmatically or by the user for

viewing, editing, or printing." Emphasis added.) This quote therefore refers to a document file stream, not a print data stream. The document file stream is merely what is read in by the Interleaf system when the Interleaf document is opened. It is not a print data stream, as recited in the claims. (Birnbaum Dec. ¶ 101).

Opening an Interleaf document is akin to opening a Word document. However, this operation does not create a print data stream. Print data are only generated when the print function is activated, e.g., a print icon in Word is clicked. (Birnbaum Dec. ¶ 102).

In order to print a document, the Interleaf reference may conceivably generate a print data stream as its output, but it does not read in a print data stream, as claimed in the '789 patent. This fundamental difference alone distinguishes Interleaf from the '789 patent. (Birnbaum Dec. ¶ 103).

C. Interleaf Does Not Parse an Input Print Data Stream

The claims recite that the input print data stream is analyzed by a parser for graphically representable objects. The Examiner purported to find this element in the Lisp interpreter operating on the Interleaf document. As stated in the portions quoted by the Examiner regarding an "active load hook" (pp. 81-82, 85), the Lisp interpreter operates on a document file stream, or, more precisely, on the Lisp objects in the document. However, as discussed above, the document file stream is not a print data stream. Therefore, even assuming the Lisp interpreter were a parser, it does not parse a print data stream. (Birnbaum Dec. ¶ 104). In addition, the Lisp interpreter (by definition) only interprets LISP scripts, and nothing else. Therefore, it is incapable of interpreting print data streams, as contrasted to the parser of the claims of the '789 Patent, which analyzes an input print data stream. (Birnbaum Dec. ¶ 105).

D. Interleaf Does Not Produce an Output Print Data Stream Based on a Transformed Input Print Data Stream

The claims recite that "(iv) the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer, and (v) the objects thus transformed are combined into an output print data stream and are output." Emphasis added.

The Examiner pointed to "transformation capabilities used to control publishing," such as automatic pagination. Although Interleaf, like many other document creation tools, may eventually create a print data stream – Interleaf does not read that stream into and analyze it using a parser and split it up into objects. See e.g., claim 1 (preamble, (i) and (ii). Not having this capability, Interleaf

obviously cannot transform objects and combine those transformed objects into an output print data stream. See e.g., Claim 1 (iv) and (v). (Birnbaum Dec. ¶ 106).

The Examiner stated "[o]pening the document for printing shows the system actually outputting the output print data stream." (Office action, p. 29). Simply, "opening" a document for printing, plainly does not: (1) read the print data stream; (2) parse it; (3) split it up into objects; (4) transform objects to control a printer; or (5) combine those transformed objects into an output data stream. (Birnbaum Dec. ¶ 107).

VI. <u>Ground of Rejection No. 6:</u> (Claims 15, 36, and 52 under 35 U.S.C. § 103(a) as obvious over the Interleaf reference in view of Lieberman. See Request pp. 13-15, and Exhibit J).

As discussed above, with reference to Ground of Rejection No. 4, Lieberman does not involve printing whatsoever. A person of ordinary skill in the relevant art would have had no reason in 2000 to combine IBM or Interleaf with Lieberman for any reason. (Birnbaum Dec. ¶ 115). Suggesting that this combination would have been made is a classic exercise of hindsight analysis, using the patent at issue as template into which to place arbitrary prior work.

The secondary indicia of non-obviousness also show why this conclusion is unsupported.

Moreover, even if Lieberman and Interleaf were combined, for reasons explained above, the combination would not result in the claimed inventions (e.g., no parsing of an input print data stream). (Birnbaum Dec. ¶ 116).

VII. <u>Grounds of Rejection No. 7:</u> (Claims 1-2, 4-14, 16-18, 22-23, 25-35, 37-39, 41-51, and 53 under 35 U.S.C. § 103(a) as obvious over the Interleaf reference in view of the Interleaf Patent. See Request p. 15, and Exhibit K.)

The Interleaf Patent does not rectify the deficiencies of the Interleaf reference disclosed above with respect to Ground of Rejection No. 5. It, too, does not relate to transforming a print data stream. To purportedly find a script executed in the cases defined in the script, the Examiner referred to the following in the Interleaf Patent: "any data object within the hierarchy may be associated with a method which, upon the occurrence of a certain event, will be implemented." (Col. 3, lines 37-39). However, this must be read in context. The paragraph reads (emphasis added):

> [A]ny data object within the hierarchy may be associated with a method which, upon the occurrence of a certain event, will be implemented. For example, referring to FIG. 5, memorandum 32 is associated with an open method 64. When a user, via user interface 30, indicates that he desires to view memorandum 32, the open method 64 is implemented to open the memorandum file 31 and prepare image data representative of how the

> document would appear if printed. The image data is supplied to display device 24 which generates an image of the document as shown in FIG. 3.

The occurrences upon which methods are implemented are in the context of user interactions with the document, e.g., user opening a document, clicking on an image, etc. They are not in a print data stream, nor are they implemented in the context of printing a document, as claimed in the '789 Patent. (Birnbaum Dec. ¶ 118).

VIII. Ground of Rejection No. 8: (Claims 15, 19, 36, and 52 under 35 U.S.C. § 103(a) as obvious over the Interleaf reference in view of the Interleaf Patent and further in view of Lieberman. Office action, p. 38).

As discussed above, neither of the Interleaf Documents, nor Lieberman involve printing whatsoever. Combining three non-printing references, as a matter of common sense, cannot create a printing invention. Further, a person of ordinary skill in the relevant art would have had no reason in 2000 to combine the Interleaf Documents with Lieberman for any reason. Suggesting that this combination would have been made is a classic exercise of hindsight analysis, using the patent at issue as template into which to place arbitrary prior work. (Birnbaum Dec. ¶ 130).

In the end, the combination is not what is claimed.

IX. Ground of Rejection No. 9: (Claims 1-53 under 35 U.S.C. § 103(a) as obvious over the Interleaf Patent in view of IBM. Office action, p. 40.)

The above remarks pertaining to the impermissible combination of the Interleaf Documents and the IBM Reference are fully applicable here.

In addition, the Examiner stated that "it is inherent that when printing, the objects of the active document to be printed are combined into an output stream." (Office action, p. 40). However, there is no overlap between the Interleaf Patent and the IBM Reference. That is, the print data stream produced by the Interleaf Patent as an output is handed off to the IBM Reference as an input. (Birnbaum Dec. ¶ 132). It makes no sense to "combine" the references in the manner suggested by Samsung, or to apply processes of the IBM Reference (print formatting and printer functionality) to those of the Interleaf Patent (document creation and editing).

Furthermore, reference in the Office action to various types of print data streams is irrelevant, because these are simply converted to AFP data streams for processing. (Birnbaum Dec. ¶ 133).

BRIEF DISCUSSION OF DEPENDENT CLAIMS

Claims 2, 23, and 39 add the limitation that objects can be combined to form super-objects. Nothing disclosed in the IBM Reference suggests that the data structures described (e.g., an image of an airplane) are linked to other objects with the properties listed above (including class hierarchy, assignment of methods to objects, and inheritance). (Birnbaum Dec. ¶ 45). Therefore, if the objects to do not possess these properties, they cannot be linked together to form super-objects with those properties. (Birnbaum Dec. ¶ 46). Samsung's contention that "'Documents' and 'pages' are examples of super-objects of higher complexity" (Exhibit G, p. 14) and graphics, made up of lines, arcs, etc. and bar codes are super-objects all fails for similar reasons: Documents, pages, lines, arcs, etc., are not objects, for reasons explained above, and combining them into a geometric shape does not create a super-objects, within the meaning of the '789 patent claims. (Birnbaum Dec.¶¶ 45-48).

Regarding Interleaf, the Examiner pointed to component objects being changed, for example, "while the document is opening" (Office action, p. 12). However, as explained above, this bears no relation to transforming a print data stream, as claimed in the '789 patent. (Birnbaum Dec. ¶ 73). Similarly, the disclosure in the reference of a template document subclass with an 'open' method used for auto-localizing documents bears no relation to transforming a print data stream. (Birnbaum Dec. ¶ 108).

Regarding the Interleaf patent, Samsung points to col. 2, line 61 to col. 3, line 13, stating that the patent discloses a nested hierarchy of data objects. (Office action, p. 19). However, as with other portions of the Interleaf patent, this bears no relation to transforming a print data stream. (Birnbaum Dec. ¶ 81).

Claims 3, 24, and 40 recite that feedback messages that indicate that the output device, e.g., the printer has recognized a transformed graphic object in the output print data stream which cannot be output by the printer. Regarding the IBM Reference (pp. 16-17 and 26), the feedback mechanism provided by IPDS does not have the functionality recited in these claims. The IBM Reference does not disclose analysis of feedback messages for error messages which indicate that the output device has recognized a transformed graphic object in the output print data stream which cannot be output by the printer, and that based on such a feedback message, splitting up the graphic object into objects of lower complexity that can be printed. (Birnbaum Dec.¶¶ 49-50).

Claims 4, 25, and 41 recite that a graphically representable object is assigned at least one script which controls external devices. The IBM Reference does not disclose scripts, scripts assigned to objects, or objects stored in object-oriented format. (Birnbaum Dec. ¶ 51). Therefore,

the IBM Reference cannot disclose that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which controls external devices, as recited. (Id.)

Regarding Interleaf, the Office action cites: (1) that active documents are defined as "structure documents. . . in which the objects. . . can be acted upon by, and can themselves act upon, other objects in the document or the outside world." (Interleaf, p. 78), and (2) that clicking on a face plays an audio name, etc. (Interleaf, p. 79). However, these portions are irrelevant to claims 4, 25, and 41, which have to do with use of objects in connection with transformation of a <u>print data stream</u>. (Birnbaum Dec. ¶ 74).

With regard to the Interleaf Patent, the Office action cites claim 9. A particular example is provided by a "phone directory that can call the person who's [sic] name you've selected; if the line is busy, it puts you into electronic mail." None of this relates to printing (there is no suggestion for the <u>printer</u> to call a person), and the AS/400's DDS would not have been capable of providing this functionality. (Birnbaum Dec. ¶ 82).

Claims 5, 26 and 42 recite that a graphically representable object is assigned at least one script which automatically receives data, "preferably data organized in an object-oriented manner, image data, text data or . . . or else e-mails."

The IBM reference does not disclose scripts, scripts assigned to objects, or objects stored in object-oriented format. (Birnbaum Dec. ¶¶ 51- 52). In addition, none of the cited portions of the IBM Reference discuss automatically receiving data organized in an object-oriented manner, for example, from the Internet, or e-mails. (Id.)

Regarding Interleaf reference's use of LISP, nothing in IBM or Interleaf suggests such use in connection with an input print data stream generally, or automatically receiving data. (Birnbaum Dec. ¶ 75).

In the Interleaf Patent, one purported application involves retrieving stock information and displaying it in text and charts; however, <u>nothing</u> suggests extracting the data <u>at print time</u>, that is, based on an input print data stream, much less doing so using the AS/400's DDS. (Birnbaum Dec. ¶ 83).

Moreover, any such capabilities in the Interleaf reference and patent are discussed with respect to the limited scope of document creation and editing. Once the document is sent to the printer, any such functionality is lost. That is, nothing in the references suggests obtaining such

information during the printing process, i.e., after receiving an input print data stream. (Birnbaum Dec. ¶ 121).

Claims 6, 27, and 43 recite that the script automatically receiving data also requests this data automatically. The cited portions of the Interleaf reference and patent do not relate to use in connection with an input print data stream. The Examiner's reference to Interleaf's documents "not appear[ing] active to all users" further reinforces the point that the Interleaf Documents relate to user editing and handling of documents, not their printing. (Birnbaum Dec. ¶¶ 76 and 84).

Claims 7, 28, and 44 recite that the script in turn reassigns the data received by it to the graphic object associated with itself, and prints out the graphic object assigned to itself together with the data requested, received and reassigned by itself.

The IBM reference does not disclose scripts. Accordingly, the functionality of the scripts recited in these claims cannot be performed with compiled IBM DDS. Therefore, even if the Interleaf reference discloses such functionality, it would not have been possible to incorporate this into the AS/400, as Samsung suggested. (Birnbaum Dec. ¶¶ 77 and 85).

The Office action points to the Interleaf Patent, which discloses a "customer receipt that automatically runs a 'frame grabber' and incorporates a photographic image of the purchaser." Again, this frame grabber in its incorporation into the receipt do not transpire based on an input print data stream. (Birnbaum Dec. ¶ 85).

Claims 8, 29, and 45 recite a one graphically representable object is assigned at least one script which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or . . . or else e-mails." The Office action cites the Interleaf Patent, including an "urgent memo that integrates with voice annotation software so when it mails itself to someone, it announces its presence audibly" and a "document created from a database which updates the database if you make any changes in the imported document." These examples have no application to printing. There is no reason to think that one of ordinary skill would have combined the references to produce an urgent memo that when printed integrates with voice annotation software so when it prints itself, it announces its presence audibly, or that this would have been possible using the tools available from the IBM Reference. (See also Birnbaum Dec. ¶¶ 86, 123).

Claims 9, 30, and 46 recite that "the script sends the graphic object associated with itself to [a] receiver." Regarding the Interleaf Patent, a "WYSIWYG document that can send itself over an electronic mail system" and "memos sent to you as a reminder of dates / events" have no application to a transformation of a print data stream, as recited in the claims. (Birnbaum Dec. ¶¶ 87, 124).

Moreover, combining Interleaf with IBM would not maintain the email functionality, for the reasons described above; and DDS is incompatible with that ability.

Claims 10, 31, and 47 recite that "the script in turn reassigns the data received by it to the graphic object associated with it, and prints out the graphic object assigned to itself together with the data requested, received and reassigned by itself." See discussion of claims 5, 26, and 42, above. (Birnbaum Dec. ¶ 88, 125).

Claims 11, 32, and 48 recite that "at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which is executed in the case of the output of the object defined in the script." The IBM Reference does not disclose scripts, but merely the use of a compiled DDS. Accordingly, the functionality of the scripts recited in these claims cannot be performed with compiled DDS. (Birnbaum Dec. ¶ 78). The Office action cited the Interleaf Patent at col. 6, lines 6-16, and col. 3, lines 37-39, but as discussed above, there is nothing to suggest combining this functionality with the IBM Reference, which does not disclose scripts, or using this functionality to transform a print data stream. (Birnbaum Dec. ¶¶ 89, 126).

Claims 12, 33, and 49 recite that a graphically representable object is assigned at least one script, at least one case relating to the execution of the script being defined in the respective script, and occurring automatically, preferably without further influence from outside. The Examiner has pointed to a number of features of the Interleaf documents, but these operate while the document is being edited or opened; they do not relate to manipulation of a print stream produced by an Interleaf document. In any event, Samsung has not shown that it would have been obvious or even possible to perform this functionality in the DDS used by the AS/400 system. (Birnbaum Dec. ¶¶ 79, 90, 114).

Claims 13, 34, and 50 recite that "that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time." **Claims 14, 35, and 51** recite that "the timer operates cyclically, that is to say it starts itself again upon expiry." The Office action cites Interleaf, e.g., a document that includes "Draft" before an announcement date, a reminder that appears on a screen based on a document due date, a weekly report built on a manager's desktop. First, these examples relate to the generation of a document, not an operation performed on a print stream based on a document. Second, simply comparing an announcement date to a current system date is not a timer function. It is not at all clear it would have been possible to

implement either the date comparison or the timer functions on the AS/400. (Birnbaum Dec. ¶¶ 91, 127).

Claims 16, 37, and 53 recite that ". . . script objects, preferably before they are output in the output print data stream, are kept ready by an application interface to be read out, to be changed, to be deleted or to have new objects appended." The Examiner referred to various editing functions disclosed in the Interleaf Patent, and stated that these are purportedly performed "before they [the objects] are output in the output print data stream" because they happen before the document is printed. This interpretation conveniently ignores that fact that the claims expressly recite that the operations are performed <u>after</u> receiving an input print data stream. Nowhere does the Interleaf Patent disclose or render obvious these functions when performed after receiving an input print data stream, as required by the claims. (Birnbaum Dec. ¶¶ 92, 128).

Claim 18 recites a system that also has an operating station with display means and input means, which makes it possible for the graphically representable objects. . . preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream. The Interleaf Patent does not disclose performing any such functions after receiving a print data stream. (Birnbaum Dec. ¶¶ 93, 129).

Claim 20 recites a "printer, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17." The Office action stated that "IBM teaches a printer . . . characterized in that it has a system for the transformation of digital print data streams." (Office action, p. 15). The AS/400 is not a printer, and a printer connected to the AS/400 would certainly not have a system for the transformation of digital print data streams – with or without the additional of any of the prior art references. Nor would it have been obvious based on either the IBM or Interleaf reference to take the functionality of the AS/400 main frame and import it into a printer. Doing so would be contrary to the centralized organization of the AS/400 system, and most likely would have been technologically impossible to have printer's microprocessor perform the operations that had been carried out by a main-frame computer. (Birnbaum Dec. ¶¶ 42-44).

The inventions of the '789 Patent are new and would not have been obvious. Indeed, as shown by ample objective evidence, large sophisticated companies such as IBM and Samsung paid millions to license the patented technology. Samsung has sold hundreds of thousands of printers with the patented software embedded, and both IBM and Samsung gave the patented software effusive praise. All originally issued claims and the newly submitted claims should be confirmed. Please charge any fees associated with this paper to deposit account No. 50-3355.

Respectfully submitted,

s/**Guy Yonay**/

Guy Yonay
Attorney/Agent for Applicant(s)
Registration No. 52,388

Dated: March 30, 2011

Pearl Cohen Zedek Latzer, LLP
1500 Broadway, 12th Floor
New York, New York 10036
Tel: (646) 878-0800
Fax: (646) 878-0801

Appendix A

APPENDIX A:

SUPPORT FOR NEW CLAIMS

In connection with the response to the Office action issued November 19, 2010, Patentee has added new claims 54-82. The claims add no new matter, are supported by the specification of the '789 patent, and do not broaden the scope of the original patent.

Claim 54 finds support throughout the specification, including the following:

Claim 54	'789 patent specification
A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates to perform the following:	"The present method according to the invention can also be present implemented on a system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, the data processing unit being programmed in such a way that it operates in accordance with an embodiment of the method according to the invention." (Col. 8, lines 1-8)
(i) read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
(ii) analyze the input print data steam by means of a parser for graphically representable objects and split up the input print data stream into these graphically representable objects;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)

Claim 54	'789 patent specification
(iii) store said graphically representable objects in the memory in an object-oriented format;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)
(iv) transform the graphically representable objects into a format for the control of a printer; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer. . ." (Col. 3, lines 5-20)
(v) combine the transformed objects into an output print data stream, and output said combined output print data stream to said printer,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)
wherein said graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in cases defined in the script.	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)

Claim 55 finds support throughout the specification, including the following:

Claim 55	**'789 patent specification**
The system according to claim 54,	See above.
characterized in that the data processing unit is further programmed to send and receive e-mails in the cases defined in the script.	"It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver. . ." (Col. 5, lines 21-26) "Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails. . ." (Col. 5, lines 38-40)

Claim 56 finds support throughout the specification, including the following:

Claim 56	**'789 patent specification**
The system according to claim 54, further comprising:	See above.
characterized in that the data processing unit is further programmed to cause original print and image data to be printed without a printer driver.	"Systems that operate on the method according to the invention, such as printing systems, are capable. . . of printing original print and image data without a printer driver." (Col. 5, lines 38-41)

Claim 57 finds support throughout the specification, including the following:

Claim 57	'789 patent specification
The system according to claim 54, further comprising:	See above.
an operating station with display means and input means, wherein said operating system makes it possible for the graphically representable objects to be read out via the application interface, to be changed, to be deleted, or to be appended before they are output in the output print data stream.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 58 finds support throughout the specification, including the following:

Claim 58	**'789 patent specification**
The system according to claim 57,	See above.
wherein said graphically representable objects to be read out, changed, deleted, or appended via the application interface are script objects.	"A further preferred embodiment of the method according to the present invention is characterized in that the graphically representable objects stored in the memory in an object-oriented format, preferably also script objects (for example Java Script objects), preferably before they are output in the output print data stream, are kept ready by an application interface to be read out, to be changed, to be deleted or to have new objects appended." (Col. 7, lines 9-16) "The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface, by assigning the methods required for this to the respective objects in accordance with their class hierarchy. This means that the objects stored in the memory can, for example, be displayed on a screen and modified as desired." (Col. 7, lines 24-32)

Claim 59 finds support throughout the specification, including the following:

Claim 59	'789 patent specification
A printer adapted for the transformation of digital print data streams comprising:	"The system according to the invention can also be integrated into a printer. . ." (Col. 8, lines 24-25)
an input to read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)
a memory to store said graphically representable objects in an object-oriented format,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)
wherein said printer is adapted to transform the graphically representable objects into a format for the control of an output device, to combine the transformed objects into an output print data stream, and to output said combined output print data stream, and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer and the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)
wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in cases defined in the script.	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)

Claim 60 finds support throughout the specification, including the following:

Claim 60	'789 patent specification
The system according to claim 59,	See above.
wherein said script is configured to operate as a timer, such that the script is executed automatically as a result of expiry of time.	"For example, the automatically occurring case, defined at least in the respective script, relating to the execution of the script can be configured as a timer, that is to say as a case which occurs automatically as a result of the expiry of a time, this timer preferably operating cyclically, that is to say starting itself again upon expiry." (Col. 6, lines 25-30).

Claim 61 finds support throughout the specification, including the following:

Claim 61	'789 patent specification
The system according to claim 59,	See above.
wherein said script when executed operates to obtain information over a network, and to use the obtained information in the output print data stream.	"Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails and of printing original print and image data without a printer driver. They are able to store any desired information on hard disks or memory cards and make said data available to all the devices connected in the network and Internet." (Col. 5, lines 38-44) "A further preferred embodiment of the method according to the invention is characterized in that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails. (Col. 5, lines 11-18)"

Claim 62 finds support throughout the specification, including the following:

Claim 62	**'789 patent specification**
The system according to claim 59,	See above.
wherein said printer is adapted to print original print and image data without a printer driver.	"Systems that operate on the method according to the invention, such as printing systems, are capable. . . of printing original print and image data without a printer driver." (Col. 5, lines 38-41)

Claim 63 finds support throughout the specification, including the following:

Claim 63	**'789 patent specification**
The system according to claim 59, further comprising:	See above.
wherein said printer is further adapted to send and receive e-mails in the cases defined in the script.	"It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver. . ." (Col. 5, lines 21-26) "Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails. . ." (Col. 5, lines 38-40)

Claim 64 finds support throughout the specification, including the following:

Claim 64	'789 patent specification
A printing system comprising	Throughout application.
a printer adapted for the transformation of digital print data streams comprising:	"The present invention relates to a method and system for the transformation of digital print data streams and corresponding printer and printer server." (Col. 1, lines 6-9)
an input to read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)
a memory to store said graphically representable objects in an object-oriented format,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)
wherein said printer is adapted to transform the graphically representable objects for the control of an output device, to combine the transformed objects into an output print data stream, and to output said combined output print data stream, and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer and the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)

Claim 64	'789 patent specification
wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in cases defined in the script; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)
an operating station with display means and input means, wherein said operating station is adapted to allow manipulation of said at least one script via an application interface.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 65 finds support throughout the specification, including the following:

Claim 65	**'789 patent specification**
The printing system according to claim 64, wherein said manipulation comprises at least one operation selected from the operations consisting of: reading out, changing, deleting, or appending said script.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 66 finds support throughout the specification, including the following:

Claim 66	**'789 patent specification**
The printing system according to claim 64, wherein said printer is adapted to print original print and image data without a printer driver.	"Systems that operate on the method according to the invention, such as printing systems, are capable. . . of printing original print and image data without a printer driver." (Col. 5, lines 38-41)

Claim 67 finds support throughout the specification, including the following:

Claim 67	'789 patent specification
The printing system according to claim 64, wherein said printer is further adapted to send and receive e-mails in the cases defined in the script.	"It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver. . ." (Col. 5, lines 21-26) "Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails. . ." (Col. 5, lines 38-40)

Claim 68 finds support throughout the specification, including the following:

Claim 68	'789 patent specification
A printer server for the transformation of digital print data streams comprising:	"The present invention relates to a method and system for the transformation of digital print data streams and corresponding printer and printer server." (Col. 1, lines 6-9)
an input to read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)

Claim 68	'789 patent specification
a memory to store said graphically representable objects in an object-oriented format,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)
wherein said printer server is adapted to transform the graphically representable objects for the control of an output device, to combine the transformed objects into an output print data stream, and to output said combined output print data stream to the output device, and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer and the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)
wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in cases defined in the script.	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)

Claim 69 finds support throughout the specification, including the following:

Claim 69	'789 patent specification
The printer server of claim 68, wherein said script is configured to operate as a timer, such that the script is executed automatically as a result of expiry of time.	"For example, the automatically occurring case, defined at least in the respective script, relating to the execution of the script can be configured as a timer, that is to say as a case which occurs automatically as a result of the expiry of a time, this timer preferably operating cyclically, that is to say starting itself again upon expiry." (Col. 6, lines 25-30).

Claim 70 finds support throughout the specification, including the following:

Claim 70	'789 patent specification
The printer server of claim 68, wherein said script when executed operates to obtain information over a network, and to use the obtained information in the output print data stream.	"Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails and of printing original print and image data without a printer driver. They are able to store any desired information on hard disks or memory cards and make said data available to all the devices connected in the network and Internet." (Col. 5, lines 38-44) "A further preferred embodiment of the method according to the invention is characterized in that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails. (Col. 5, lines 11-18)"

Claim 71 finds support throughout the specification, including the following:

Claim 71	'789 patent specification
The printer server of claim 68, wherein said output device is a printer.	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer[.]" (Col. 3, lines 5-14)

Claim 72 finds support throughout the specification, including the following:

Claim 72	'789 patent specification
A printing system comprising	Throughout application.
a printer server adapted for the transformation of digital print data streams comprising:	"The present invention relates to a method and system for the transformation of digital print data streams and corresponding printer and printer server." (Col. 1, lines 6-9)
an input to read an input print data stream;	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in. . ." (Col. 3, lines 5-20)
a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . this [input print data stream] is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects. . ." (Col. 3, lines 5-20)
a memory to store said graphically representable objects in an object-oriented format,	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects are stored in a memory in an object-oriented format. . ." (Col. 3, lines 5-20)
wherein said printer server is adapted to transform the graphically representable objects into a format for the control of a printer, to combine the transformed objects into an output print data stream, and to output said combined output print data stream to the printer, and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which. . . the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer and the objects thus transformed are combined into an output print data stream and are output. . ." (Col. 3, lines 5-20)

Claim 72	'789 patent specification
wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in cases defined in the script; and	"According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which . . . graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script." (Col. 3, lines 5-20)
an operating station with display means and input means, wherein said operating station is adapted to allow manipulation of said at least one script via an application interface.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 73 finds support throughout the specification, including the following:

Claim 73	'789 patent specification
The printing system according to claim 72, wherein said manipulation comprises at least one operation selected from the operations consisting of: reading out, changing, deleting, or appending, said script.	"The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface. . ." (Col. 7, lines 24-28) "In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream." (Col. 8, lines 9-16)

Claim 74 finds support throughout the specification, including the following:

Claim 74	'789 patent specification
The printing system according to claim 72, wherein said printer server is adapted to print original print and image data on the printer without a printer driver associated with the printer.	"Systems that operate on the method according to the invention, such as printing systems, are capable. . . of printing original print and image data without a printer driver." (Col. 5, lines 38-41)

Claim 75 finds support throughout the specification, including the following:

Claim 75	**'789 patent specification**
The system according to claim 72, wherein said printer server is further adapted to send and receive e-mails in the cases defined in the script.	"It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver. . ." (Col. 5, lines 21-26) "Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails. . ." (Col. 5, lines 38-40)

Claim 76 finds support throughout the specification, including the following:

Claim 76	**'789 patent specification**
The method of claim 1, wherein said input data stream is formatted in a page description language.	"In this case. . . the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used." (Col. 3, lines 21-25)

Claim 77 finds support throughout the specification, including the following:

Claim 77	**'789 patent specification**
The method of claim 76, wherein said parser is a syntax analyzer, and wherein said analyzing by means of said parser comprises performing syntactic analysis on said input data stream.	"In this case. . . the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used." (Col. 3, lines 21-26)

Claim 78 finds support throughout the specification, including the following:

Claim 78	'789 patent specification
The method of claim 1, wherein storing the graphically representable objects in said object-oriented format comprising storing said graphically representable objects based on membership in hierarchically organized classes.	"The individual graphic objects are stored by using their membership of specific – expediently suitably hierarchically organized – classes. . . " (Col. 3, lines 41-44)

Claim 79 finds support throughout the specification, including the following:

Claim 79	'789 patent specification
The method of claim 1, further comprising dynamically linking a plurality of objects.	"In relation to the above explanations, it should be noted that the embodiments of the method according to the invention which themselves provide other objects with objects, for example by forwarding them or keeping them ready to receive or for interrogation by a script, for example, are also covered by the term 'dynamic object linking' (DOL)." (Col. 5, lines 32-37)

Claim 80 finds support throughout the specification, including the following:

Claim 80	'789 patent specification
The method of claim 1, wherein the objects are managed by display list management module.	"The objects are preferably managed here by means of a "display list management", which supports one-page and multi-page documents at as many levels as desired and which can be expanded dynamically by new objects." (Col. 3, lines 38-41)

Claim 81 finds support throughout the specification, including the following:

Claim 81	'789 patent specification
The method of claim 80, wherein the display list management supports one page and multi-page documents at a plurality of levels.	"The objects are preferably managed here by means of a "display list management", which supports one-page and multi-page documents at as many levels as desired and which can be expanded dynamically by new objects." (Col. 3, lines 38-41)

Claim 82 finds support throughout the specification, including the following:

Claim 82	'789 patent specification
The method of claim 81, wherein the display list management can be expanded dynamically by new objects.	"The objects are preferably managed here by means of a "display list management", which supports one-page and multi-page documents at as many levels as desired and which can be expanded dynamically by new objects." (Col. 3, lines 38-41)

Appendix B

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF NEW JERSEY

CCP SYSTEMS AG, Plaintiff, v. SAMSUNG ELECTRONICS CORP., LTD., SAMSUNG ELECTRONICS AMERICA, INC., SAMSUNG NETWORKS, INC. and IBM CORPORATION, Defendants.	Civil Action No: Jury Trial Demanded

VERIFIED COMPLAINT FOR COPYRIGHT AND PATENT INFRINGEMENT

CCP Systems AG ("CCP") for its Verified Complaint against Samsung Electronics Corp., Ltd. ("Samsung ECL"), Samsung Electronics America, Inc. ("Samsung America"), Samsung Networks, Inc. ("Samsung Networks"), and IBM Corp. ("IBM") (collectively, "Defendants"), states as follows:

PARTIES

1. CCP is a corporation organized and existing under the laws of Germany with its principal place of business at Stammheimer Str. 35, 70435 Stuttgart, Germany.

2. Upon information and belief, Defendant Samsung ECL is a corporation organized and existing under the laws of South Korea, with its principal place of business at Samsung Main Bldg. 250, 2-Ga, Taepyung-Ro Joong-Gu, Seoul, Korea, 100742.

3. Upon information and belief, Defendant Samsung America is a corporation organized and existing under the laws of the State of New York, with its principal place of business at 105 Challenger Road, Ridgefield Park, New Jersey, 07660.

4. Upon information and belief, Defendant Samsung Networks is a corporation organized and existing under the laws of South Korea, with its principal place of business at 9th Floor, Asem Tower, Samseong 1(il)-Dong , Gangnam-Gu Seoul, Korea, 135798.

5. Upon information and belief, Defendant IBM is a corporation organized and existing under the laws of the State of New York, with its principal place of business at 1 New Orchard Road, Armonk, NY 10504.

JURISDICTION AND VENUE

6. This is an action for copyright infringement under the Copyright Act, 17 U.S.C. § 106, and for patent infringement under the Patent Act, 35 U.S.C. §§ 271 and 281-285.

7. This Court has subject matter jurisdiction over the Federal law claims in this case pursuant to 28 U.S.C. §§ 1331 and 1338(a)-(b).

8. Venue is proper in this District pursuant to 28 U.S.C. §§ 1391 and 1400 because the Defendants may be found in this District, do substantial business in this District, and have engaged in acts of copyright and patent infringement in this District.

9. This Court has personal jurisdiction over Defendant Samsung America because New Jersey is Samsung America's principal place of business.

10. This Court has personal jurisdiction over Defendant IBM by virtue of IBM's transacting business within this State, and because of its systematic and continuous contacts with residents of this State.

11. This Court has personal jurisdiction over Defendants Samsung ECL and Samsung Networks by reason of those Defendants transacting business within this State and committing wrongful acts within this State, and by virtue of their systematic contacts with residents of this State.

12. In particular, this Court has personal jurisdiction over Defendants Samsung ECL and Samsung Networks because they operate and maintain interactive websites which can be accessed, and, upon information and belief, have been accessed, by residents of this State.

FACTS COMMON TO ALL CLAIMS

A. CCP's Software Products

13. CCP designs, develops, manufactures, sells and distributes software for use in connection with computer printers and other computer-related devices.

14. In particular, and relevant to this action, CCP developed software called "JScribe®," which is an open application and communication platform for servers, workstations, multi-function printers, and standard printers, allowing those devices to exchange information proactively in every direction.

15. The JScribe software has been the basis for many additional programs and variations. CCP has designed and developed those additional programs and variations into several applications, including the following:

a. "JScribe Core," which is the version of JScribe for embedding on individual printers, multi-function printers, and other devices.

b. "JScribe Mobile Print Solution" ("JMPS"), which is a JScribe-based client/server solution for mobile printing in company and public networks.

c. "JScribe Software Developer Kit" ("JSDK"), which is a JScribe-based integrated development environment for the development of JScribe applications.

d. "JISS OpenPower," which stands for "JScribe Intelligence Server Solution OpenPower," which is a server-based solution for tracking print and managing related costs, implemented on Linux/Unix platforms according to requirements defined by IBM.

f. "JTalk" is a JScribe-based tool for deploying JScribe applications to printers and other devices with JScribe Core or the JScribe Application Server Solution ("JASS") embedded.

g. "KYAOC" is a custom derivative of JMPS developed by CCP for a specific client, the State of Kentucky Administrative Office of Courts.

(The JISS OpenPower, JMPS, JSDK, JScribe Core, JTalk and KYAOC software are referenced collectively throughout this Complaint as the "CCP Software.")

B. <u>The Limited Licenses Granted to Defendants by CCP</u>

16. In 2004, CCP signed a contract with IBM Deutschland GmbH ("IBM Germany") (the "IBM Germany Agreement"), under which, *inter alia*, (a) CCP would license and deliver copies of JScribe Core to IBM Germany, and (b) IBM Germany then would sublicense and deliver copies of the JScribe Core software to its customers, solely to be bundled with other software in the customer's "Firmware" and embedded into the customer's device, such as a printer. ("Firmware" is a microprogram stored in ROM [read-only-memory], designed to implement a function that had previously been provided in software.) Under the IBM Germany Agreement, no other uses of the JScribe Core software were permitted by IBM Germany or its customers.

17. Upon information and belief, Samsung ECL, under a written sublicense agreement with IBM Korea (IBM Germany's Korean sister company, which, upon information and belief, has a sublicense arrangement with IBM Germany), embedded the JScribe Core software with other software of Samsung ECL, creating "Firmware," which it would then in turn embed directly into a Samsung-brand device, such as a printer.

18. According to the IBM Germany Agreement, as an IBM customer with an authorized sublicense, Samsung ECL, for itself and through its affiliates, such as Samsung America, could distribute devices containing the Firmware (which included the JScribe Core) to consumers in

the United States and elsewhere. Samsung would then pay a royalty to IBM Germany (or its sister company in Korea), who would then in turn pay a royalty to CCP for each device sold that incorporated the JScribe Core software.

19. The IBM Germany Agreement did not provide that an IBM customer, such as Samsung ECL, could distribute the JScribe Core software (or any other CCP Software) apart from a device, or to make any CCP Software publicly available online.

C. The IBM Agreement is Terminated

20. The IBM Germany Agreement, as amended, allowed CCP to terminate that agreement without further notice if certain events transpired. In particular, the IBM Germany Agreement allowed CCP to terminate the Agreement if CCP and IBM Germany could not reach agreement as to CCP's proposed business relationship with a competitor of IBM Germany. Since CCP and IBM Germany were not able to reach resolution on a proposed relationship between CCP and Samsung (a competitor of IBM Germany), CCP, on May 25, 2009 (the "Termination Date"), sent notice to IBM Germany that as of the Termination Date, CCP was terminating the IBM Germany Agreement and the license to the CCP Software granted therein.

21. Samsung ECL acknowledged the termination of the IBM Germany Agreement in an email dated July 15, 2009. In that email, Mr. Chin Yoon, the Vice President of Samsung ECL's Solution Business Group, instructed Samsung ECL personnel to refrain from marketing devices incorporating CCP Software, and notified them that the agreement under which the CCP Software was supplied to Samsung ECL had been cancelled.

22. Although CCP terminated the IBM Germany Agreement and the license therein to CCP Software as of the Termination Date of May 25, 2009, the Defendants, without CCP's consent,

have continued to reproduce and publicly distribute the CCP software, both as stand-alone software (bundled into Firmware) and in devices.

23. In addition, without CCP's consent, Samsung ECL, Samsung Networks, and Samsung America have placed the CCP Software, including the Firmware incorporating JScribe Core software, online at Samsung Network's website "downloadcenter.samsung.com" and "samsungprinter.info," available for free to the public.

COUNT I - COPYRIGHT INFRINGEMENT: DEFENDANT SAMSUNG NETWORKS

24. CCP incorporates the allegations of paragraphs 1 through 23 above as though fully set forth herein.

25. Samsung Networks owns and operates the website samsung.com. The samsung.com website has many sub-domains, including downloadcenter.samsung.com (the "Download Center Website").

26. Any United States resident with an Internet connection can access the Download Center Website, so long as that person knows the URLs for the code files on that website; no password or other security key is required to access it.

27. Samsung Networks has reproduced, publicly displayed, and, upon information and belief, publicly distributed the JScribe Core software on the Download Center Website, as part of the Firmware, and this conduct continues as of the date this Complaint was filed.

28. Anyone, including New Jersey residents, can download the Firmware, incorporating the JScribe Core software, for free, from the Download Center Website, if the person has the URL links to the Firmware files on the Download Center Website.

29. CCP did not authorize Samsung Networks or anyone else to make its JScribe Core software publicly available online.

30. CCP developed the JScribe Core software over the course of ten years, and that software program is an original work of authorship.

31. CCP has registered claims to copyrights in four versions of the JScribe Core software (JScribe Core v. 4.0, JScribe Core v. 4.1, JScribe Core v. 4.2, and JScribe Core v. 4.3) with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-843 to the JScribe Core v 4.0 claim to copyright, registration number TXu-1-610-827 to the JScribe Core v 4.1 claim to copyright, registration number TXu-1-610-915 to the JScribe Core v 4.2 claim to copyright, and registration number TXu-1-610-918 to the JScribe Core v 4.3 claim to copyright. These four versions will be referenced collectively throughout this Complaint as "JScribe Core."

32. By reproducing and publicly displaying the JScribe Core software at the Download Center Website, and by publicly distributing it on that site, Samsung Networks has directly infringed CCP copyrights in the JScribe Core software.

33. Samsung Networks knew that i) making the JScribe Core software available online, apart from devices, was not authorized under any agreement with CCP; and ii) in any event, as of May 25, 2009, all licenses to the JScribe Core software had terminated. Therefore, Samsung Networks' making the JScribe Core software available online for free download constitutes a willful infringement of CCP's copyrights.

34. CCP has suffered irreparable harm because of Samsung Networks' infringement of its copyrights in the JScribe Core software, and will continue to suffer irreparable harm in the future unless the Defendants are enjoined from infringing CCP's copyrights in the JScribe Core software.

35. CCP has suffered damages as a result of Samsung Networks' infringing conduct.

WHEREFORE, CCP requests the following relief against Samsung Networks:

a) an award of compensatory damages and disgorgement of any profits of Samsung Networks attributable to the infringement, along with prejudgment interests and costs;

b) a preliminary and permanent injunction prohibiting Samsung Networks and its officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing CCP's copyrights in the JScribe Core software, specifically prohibiting Samsung Networks from i) making the JScribe Core software available on the Download Center Website, ii) displaying or distributing any links to the Download Center Website; and iii) otherwise reproducing, displaying, distributing or preparing derivatives of the CCP Software;

c) a Court order requiring Samsung Networks to impound any infringing articles in its possession, custody, or control; and

d) such other relief as this Court deems just and proper.

COUNT II - CONTRIBUTORY AND VICARIOUS COPYRIGHT INFRINGEMENT: DEFENDANTS SAMSUNG ECL AND SAMSUNG AMERICA

36. CCP incorporates the allegations of paragraphs 1 through 35 above as though fully set forth herein.

37. Samsung ECL owns and operates a website at the URL samsungelectronics.com. That website has many sub-domains, including the sub-domain ecms.samsungelectronics.com (the "Partner Site").

38. Upon information and belief, Samsung ECL operates the Partner Site for the benefit of its dealers, developers and distributors worldwide, including in New Jersey.

39. To access content on the Partner Site, one must be registered with Samsung ECL, and Samsung ECL must approve, each application for access to the Partner Site.

40. Both before and after May 25, 2009, Samsung ECL made a spreadsheet available on the Partner Site. That spreadsheet contains information about various Samsung ECL products, including the Firmware that incorporates CCP's JScribe Core software.

41. That spreadsheet contains active hyperlinks to the Download Center Website. When one clicks on a hyperlink in the spreadsheet, one is taken directly to a ZIP file on the Download Center Website, where one can freely access and download the Firmware containing CCP's JScribe Core software.

42. The spreadsheet described in paragraphs 40 and 41 is downloadable and transferable. That is, anyone accessing that spreadsheet on the Partner Site can save that spreadsheet to his computer, copy it, attach it to an email, forward it to others, etc.

43. Samsung America controlled the website samsungprinter.info (the "Samsung Printer Website"). The Samsung Printer Website was open to the public.

44. Samsung America placed a spreadsheet similar to the one described in paragraphs 41 and 42 above on the Samsung Printer Website, containing links to the code files on the Download Center Website.

45. No password or other security information was required to access the code files on the Download Center Website, so anyone with this spreadsheet could freely access the code files in the Download Center Website.

46. The unauthorized availability of JScribe Core on the websites described above materially impacts the market for this software and CCP's ability to exploit its product. As an example of the effect on the market for the Firmware, CCP discovered an inquiry on a developer chat website, "Fix Ya," attached as Exhibit A. In that post, a web user asked whether anyone knew where to find the Firmware for a Samsung-brand printer with JScribe 4.0 embedded. Another user replied, identifying himself as a developer, and stated that if the user could not acquire the Firmware on the samsung.com website, he would get the Firmware from the Partner Site and send it to him. This example illustrates how Samsung ECL's, Samsung America's, and Samsung Networks' posting of the Firmware code files online has resulted in uncontrolled distribution of CCP's software products.

47. By posting the spreadsheet containing links to the Download Center Website online at the Partner Site and the Samsung Printer Website, Samsung ECL and Samsung America were aware of and supervised, facilitated and induced the direct infringement by Samsung Networks alleged in Count I. Further, by providing the means by which that infringement can occur, *inter alia*, Samsung ECL and Samsung America substantially participated in Samsung Networks' direct infringement. Accordingly, Samsung ECL and Samsung America contributorily infringed CCP's copyrights in the JScribe Core software.

48. In addition, by facilitating Samsung Networks' direct infringement, Samsung ECL and Samsung America garner goodwill with their partners, developers and distributors by making various products available on the Download Center Website and the Samsung Printer Website, and they also gain financial benefit by avoiding paying a royalty for use of the CCP Software. Accordingly, Samsung ECL and Samsung America have vicariously infringed CCP's copyrights in the JScribe Core software through their supervision and control of Samsung Networks.

49. CCP has suffered damages as a result of the infringing conduct of Samsung ECL and Samsung America.

50. CCP has suffered irreparable harm because of Samsung ECL's and Samsung America's contributory and vicarious infringement of its copyrights in the JScribe Core software, and will continue to suffer irreparable harm in the future unless Samsung ECL and Samsung America are enjoined from infringing CCP's copyrights in the JScribe Core software.

WHEREFORE, CCP requests the following relief against Samsung ECL and Samsung America:

a) an award of compensatory damages and disgorgement of any profits of Samsung ECL and Samsung America attributable to the infringement, along with prejudgment interests and costs;

b) a preliminary and permanent injunction prohibiting Samsung ECL, Samsung America, and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them from infringing CCP's copyrights in the JScribe Core software, specifically, prohibiting Samsung ECL and Samsung America from i) making the JScribe Core software available on the Download Center Website, ii) displaying or distributing any links to the Download Center Website; and iii) otherwise reproducing, displaying, distributing or preparing derivatives of the CCP Software;

c) a permanent injunction prohibiting Samsung ECL, Samsung America, and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and

all those controlled by or acting in concert with or in privity with any of them from infringing CCP's copyrights in the JScribe Core software, specifically, prohibiting Samsung ECL and Samsung America from i) making links to the CCP Software on the Download Center Website available on other websites such as the Samsung Printer Website, and ii) otherwise reproducing, displaying, distributing or preparing derivatives of the CCP Software;

d) a Court order requiring Samsung ECL and Samsung America to impound any infringing articles in their possession, custody, or control; and

e) such other relief as this Court deems just and proper.

COUNT III - COPYRIGHT INFRINGEMENT (DIRECT, CONTRIBUTORY AND VICARIOUS): DEFENDANT SAMSUNG AMERICA

51. CCP incorporates the allegations of paragraphs 1 through 50 as though fully set forth herein.

52. CCP has registered a claim to copyright in the JMPS software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-843 to the JMPS claim to copyright.

53. CCP has registered a claim to copyright in the JSDK software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-846 to the JSDK claim to copyright.

54. CCP has registered a claim to copyright in the JTalk software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-862 to the JMPS claim to copyright.

55. CCP has registered a claim to copyright in the KYAOC software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-844 to the JSDK claim to copyright.

56. Upon knowledge and belief, Samsung America placed copies of the CCP Software, including JScribe Core, JMPS, JSDK, JTalk and KYAOC, on the Samsung Printer Website, available for free download. In particular, Samsung America placed source code files for KYAOC on the Samsung Printer Website without authorization.

57. In addition, the Samsung Printer Website contained files with detailed instructions for how to use the CCP Software found on that website to install CCP Software onto hardware and other devices. These detailed instructions facilitated and encouraged unauthorized reproduction of the CCP Software by third parties, for which Samsung America garnered profit, thus, making those instructions available on the Samsung Printer Website constitutes contributory and vicarious infringement.

58. CCP never consented to or authorized the reproduction and distribution of its software products on the Samsung Printer Website.

59. CCP has suffered damages as a result of this infringing conduct.

60. CCP has suffered irreparable harm because of Samsung America's infringement of its copyright in the CCP Software, and will continue to suffer irreparable harm in the future unless Samsung America are enjoined from infringing CCP's copyrights in the CCP Software.

WHEREFORE, CCP requests the following relief against Samsung America:

a) an award of compensatory damages and disgorgement of any profits of the Defendants attributable to the infringement, along with prejudgment interests and costs;

b) a permanent injunction prohibiting Defendant Samsung America and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing CCP's copyrights in the CCP Software, specifically, by prohibiting Defendants from reproducing and distributing the CCP Software on other websites, such as the Samsung Printer Website, and by otherwise reproducing, distributing, displaying or preparing derivatives of the CCP Software;

c) a Court order requiring Samsung America to impound any infringing articles in its possession, custody, or control; and

d) such other relief as this Court deems just and proper.

COUNT IV - COPYRIGHT INFRINGEMENT: DEFENDANTS SAMSUNG ECL AND SAMSUNG AMERICA

61. CCP incorporates the allegations of paragraphs 1 through 60 as though fully set forth herein.

62. Samsung ECL reproduced CCP Software, namely, the JScribe Core software, in the process of creating the Firmware. Although the parties' agreements contemplated distribution of the Firmware (and with it, the CCP software) only embedded within a printer or other device, Samsung ECL bundled the Firmware for independent distribution on portable media, such as CDs or external hard drives, as well as by email distribution in code files. This bundling involved reproduction of CCP's JScribe Core software.

63. Once bundled and uploaded to such portable media or prepared for email distribution, Samsung ECL distributed that Firmware (including the JScribe Core software) to customers in the United States, through its US affiliate Samsung America in New Jersey.

64. Samsung ECL and Samsung America distributed some Firmware to customers who had already purchased a device containing an earlier version of the JScribe Core software. This Complaint will refer to this delivery as a "field upgrade."

65. Samsung ECL and Samsung America also distributed the Firmware to customers who had already purchased a device that did not contain any version of the JScribe Core software. This Complaint will refer to this delivery as a "field installation."

66. With each "field installation" and "field upgrade," Samsung ECL and Samsung America provided the customer with detailed instructions showing the customer how to install the Firmware onto an existing Samsung-brand device. These instructions effectively allow a customer to obtain the JScribe functionality without purchasing a JScribe-embedded device. Upon knowledge and belief, Samsung America and Samsung ECL have performed field installations and field upgrades for customers.

67. In addition to directly distributing the Firmware to customers, Samsung ECL and Samsung America made the Firmware (including the JScribe Core software) available on the Download Center Website and the Samsung Printer Website, along with detailed instructions showing a customer (or other web user) how to install the Firmware onto a Samsung-brand printer to achieve the functionality of a Samsung-brand printer with JScribe Core embedded.

68. CCP never consented to or authorized the reproduction and distribution of its software products as "field installations" and "field upgrades."

69. CCP has suffered damages as a result of this infringing conduct.

70. CCP has suffered irreparable harm because of Samsung ECL's and Samsung America's infringement of its copyright in the JScribe Core software, and will continue to suffer irreparable harm in the future unless Samsung ECL and Samsung America are enjoined from infringing CCP's copyright in the JScribe Core software.

WHEREFORE, CCP requests the following relief against Samsung ECL and Samsung America:

a) an award of compensatory damages and disgorgement of any profits of the Defendants attributable to the infringement, along with prejudgment interests and costs;

b) a permanent injunction prohibiting Defendants Samsung ECL and Samsung America and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing CCP's copyrights in the JScribe Core software, specifically, by prohibiting these Defendants from reproducing and distributing the JScribe Core software via the "field installations" and "field upgrades" as described herein, and by otherwise reproducing, distributing, displaying or preparing derivatives of the CCP Software;

c) a Court order requiring said Defendants to impound any infringing articles in their possession, custody, or control; and

d) such other relief as this Court deems just and proper.

COUNT V - COPYRIGHT INFRINGEMENT: DEFENDANT IBM

71. CCP incorporates the allegations of paragraphs 1 through 70 above as though fully set forth herein.

72. CCP has registered a claim to copyright in the JISS OpenPower software with the United States Copyright Office. The Copyright Office has assigned registration number TXu-1-610-847 to the JISS OpenPower claim to copyright.

73. As noted above, CCP terminated the IBM Germany Agreement on May 25, 2009. Nevertheless, prior to May 25, 2009, and at least as early as December, 2006, the license to the JISS OpenPower software in the IBM Germany Agreement had expired by its own terms.

74. Upon information and belief, IBM has reproduced the JISS OpenPower in the process of selling servers and other devices, after December of 2006, and in any event after May 25, 2009. This reproduction of JISS OpenPower constitutes infringement of CCP's copyrights in the JISS OpenPower software.

75. Since December 2006, IBM has continued to market and, upon information and belief, sell products incorporating the JISS OpenPower software. For example, price lists and descriptions of products incorporating JISS OpenPower are available (at least as of August 7, 2009) on IBM's website. See Exhibit C.

76. Upon information and belief, IBM has publicly distributed the JISS OpenPower software in the United States, after December 2006. This public distribution of JISS OpenPower constitutes infringement of CCP's copyrights in the JISS OpenPower software.

77. CCP has suffered damages as a result of this infringing conduct.

78. CCP has suffered irreparable harm by IBM's infringement of CCP's copyrights in the JISS OpenPower software, and will continue to suffer irreparable harm in the future unless IBM is enjoined from infringing CCP's copyrights in that work.

WHEREFORE, CCP requests the following relief against IBM:

a) an award of compensatory damages and disgorgement of any profits of IBM attributable to the infringement, along with prejudgment interests and costs;

b) permanent injunctive relief prohibiting IBM and its officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing CCP's copyrights in the JISS OpenPower software, specifically, by prohibiting IBM from reproducing and distributing the JISS OpenPower software in devices, and by otherwise reproducing, distributing, displaying or preparing derivatives of the JISS OpenPower software;

c) a Court order requiring IBM to impound any infringing articles in its possession, custody, or control; and

d) such other relief as this Court deems just and proper.

COUNT VI - PATENT INFRINGEMENT (U.S. PATENT NO. 6,684,789) AGAINST SAMSUNG AMERICA, SAMSUNG ECL, AND SAMSUNG NETWORKS

79. CCP incorporates the allegations of paragraphs 1 through 78 above as though fully set forth herein.

80. On February 3, 2004, the United States Patent and Trademark Office (USPTO) duly and legally issued U.S. Patent No. 6,684,789 ("the '789 Patent") entitled "Method and System for the Transformation of Digital Print Data Streams and Corresponding Printer and Printer Server," was duly and legally issued to CCP, as assignee of inventor Thomas Krautter (a CCP employee). A copy of the '789 Patent is attached hereto as Exhibit B.

81. Samsung ECL, Samsung America, and Samsung Networks have been and are infringing, inducing infringement and/or contributing to infringement of the '789 Patent in this District, and throughout the United States, by making, selling, offering for sale, and/or importing infringing devices, software and other technology covered by one or more claims of the '789 Patent, including at least Samsung ECL's printer devices incorporating the JScribe Core technology.

82. As a direct and proximate result of the Defendants' infringement of the '789 Patent, CCP has suffered and continues to sustain monetary damages.

83. CCP has been and continues to be irreparably harmed by the Defendants' infringement of the '789 Patent. On information and belief, the Defendants will continue to infringe unless such infringement is enjoined by this Court.

WHEREFORE, CCP requests the following relief against the Defendants:

a) an award of compensatory damages and disgorgement of any profits of the Defendants attributable to the infringement, along with prejudgment interests and costs;

b) a permanent injunction prohibiting Defendants and their officers, agents, divisions, affiliates, subsidiaries, employees, and representatives, and all those controlled by or acting in concert with or in privity with any of them, from infringing, inducing the

infringement and/or contributing to the infringement of the '789 Patent pursuant to 35 U.S.C. §283; and

c) such other relief as this Court deems just and proper.

DEMAND FOR JURY TRIAL

Plaintiff CCP demands a jury trial on all issues.

Dated: August 25, 2009

Respectfully submitted,

SONNENSCHEIN NATH & ROSENTHAL LLP

By: s/ Marc S. Friedman

Marc S. Friedman
101 JFK Parkway
Short Hills, NJ 07078

and

1221 Avenue of the Americas
New York, NY 10020

212.768.6700
Fax: 212.768.6800
mfriedman@sonnenschein.com
bdelfin@sonnenchein.com
rstroder@sonnenschein.com

Attorneys for Plaintiff CCP Systems AG

Of Counsel:
Benito Delfin, Jr.
Rebecca Stroder

VERIFICATION OF CHRISTOPH PICHT UNDER 28 U.S.C. § 1746

I have read the foregoing Complaint. I have personal knowledge of the truth of the allegations contained therein, except for those allegations made upon information and belief.. For those allegations made upon information and belief, I believe them to be true.

I declare, under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed the 25th day of August, 2009

Christoph Picht

EXHIBIT A

FixYa Technical Support, Instructions, & Repair Service

Home | My Expertise | Sign In

Find a Solution Search

All Products > Samsung > SCX-6345N Printer > Troubleshooting

I need to know where obtain the firmware to

mazalazar on Jun 24, 2009

I need to know where obtain the firmware to SCX-6345 with Jscribe 4.o included

Comments:

Jun 25, 2009- Komguy
thank

I Can Solve This! I have a similar problem Post a new problem

Best Solution
posted on Jun 24, 2009

komguy
Rank: Guru
Rating: 89%, 358 votes

If you can't get this firmware at Samsung.com, let me know, I can get it from the service site and e-mail it to you.

Was this helpful? Yes **No** 1 person thought this was helpful

Solution #2
posted on Jul 21, 2009

msalvadorcan
Rank: Apprentice
Rating: 0%, 0 votes

If you do not have the corresponding NIC-Firmware it will not work. Thats the problem i have currently.

Was this helpful? Yes **No**

Need Parts & Services?

- Chat Live with a Top Printer Expert
- Find the Top Printer Repairers in your area

Solve this problem

Do you have a solution to this problem?

Share your knowledge with the FixYa Community today!

Post Solution

Related Problems

gyrene9940 just asked: Trying to print from desk to. wants to scan. - Solve this!

Ask our Experts

Describe your Samsung SCX-6345N Printer Problem

Get Immediate Assistance! Ask

Chat with a Pro

Printers Tech
Chat Live Now!

More Common Problems

For Samsung SCX-6345N Printer:
Samsung SCX-6345N Printer

Related Products & Issues:
6345 jsc JScribe samsung scx 6345 firmware

Top Printer Experts

adirondacktr
Rank: Guru
Rating: 86.0%, 155 votes
Solutions: 431
Member Since: July 2009

Experience: hands on experience,training, mechanically gifted,and favorite reading material is how to fix things

Ask Me

Find more Printer Experts

Solve Your Problem Now!
Chat Live with an Expert
Chat Now

Can you Solve these Problems?

Printers

printer program 1 Solution

Gday, I have installed printer drivers for a Canon Pixma ip4200 with no problems. On...

Printers

LEXMARK X1240 All In One 1 Solution

Will not use color cartridge and pisplay statis shows the cartridge is full...

Printers

Unable to print or copy Brother MFC-440CN 1 Solution

I am unable print, copy from this unit. The message is Unable to print 8F. There is...

Printers

I have the pixma ip5300 printer and I need... 1 Solution

I love this printer. It has six ink cartridges. After many printings on printable...

Can you Help with these Printer problems?

Printers

need to have printer down loaded to... Solve

I need to have a printer driver down load for Lexmark printer model # 4126-003. I have windows XP... (More)

Printers

error 0xc18a0301 ink system failure Solve

error 0xc18a0301 ink system failure

Printers

Changed both color and b/w... Solve

Changed both color and b/w cartridge. Now anything printed off internet prints in pink color only... (More)

Printers

I have a dell 948 printer that will... Solve

I have a dell 948 printer that will not scan. When I hit "scan to computer" it says "downloading... (More)

Samsung SCX-6345N...

NIC-Firmware for SCX 6345 Solve

Hello, I would need the corresponding NIC-Firmware for JScribe enabled SCX 6345N. Could someone help... (More)

gyrene9940 just asked: Trying to print from desk to. wants to scan. - Solve this!

Can you Solve these Problems?

Didn't find what you were looking for?

Describe your problem:

Post Problem

Get Immediate Assistance!

Solutions to Most Common SCX-6345N Printer Problems
See other Samsung Printers
See other Samsung Products
See other Printers
All Brands

Browse Experts
Find Products
Find Cars
Find Manuals
Repair Service Directory
Ratings & Recommendations
Tags

About FixYa
News
Contact
Terms
Privacy Policy

Blog
Advertisers
Partners
Careers
FAQ

Find a Solution Search

gyrene9940 just asked: Trying to print from desk to. wants to scan. - Solve this!

Can you Solve these Problems?

EXHIBIT B

US006684789B2

(12) **United States Patent**
Krautter

(10) Patent No.: **US 6,684,789 B2**
(45) Date of Patent: **Feb. 3, 2004**

(54) METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER

(75) Inventor: Thomas Erfinders Krautter, Stuttgart (DE)

(73) Assignee: CCP Systems AG, Stuttgart (DE)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: 10/275,784

(22) PCT Filed: May 11, 2001

(86) PCT No.: PCT/DE01/01796

§ 371 (c)(1),
(2), (4) Date: Nov. 7, 2002

(87) PCT Pub. No.: WO01/88840

PCT Pub. Date: Nov. 22, 2001

(65) Prior Publication Data

US 2003/0140809 A1 Jul. 31, 2003

(30) Foreign Application Priority Data

May 17, 2000 (DE) 100 24 177
Jan. 26, 2001 (DE) 101 03 733

(51) Int. Cl.[7] B41F 1/54
(52) U.S. Cl. 101/484; 101/486; 400/61; 400/62
(58) Field of Search 101/484, 485, 101/486; 400/61, 62, 63, 76; 358/1.1, 1.9, 1.15, 1.16, 1.17, 1.18

(56) References Cited

U.S. PATENT DOCUMENTS

5,216,754 A 6/1993 Sathi et al.
5,566,278 A * 10/1996 Patel et al. 358/1.15
6,006,013 A 12/1999 Rumph et al.

FOREIGN PATENT DOCUMENTS

EP	0 109 615 B1		6/1994
EP	0 964 339 A2		12/1999
EP	1 061 456 A2		12/2000
GB	2 357 348 A		6/2001
WO	WO-00/17748		3/2000
WO	WO-00/55720		9/2000

* cited by examiner

Primary Examiner—Andrew H. Hirshfeld
Assistant Examiner—Minh Chau
(74) *Attorney, Agent, or Firm*—Squire, Sanders & Dempsey L.L.P.

(57) ABSTRACT

A method for the transformation of digital print data streams, in which an input print data stream (2) is read in, this is analyzed by means of a parser (4) for graphically representable objects (5, 5*a*) and is split up into these graphically representable objects (5, 5*a*), and the graphically representable objects (5, 5*a*) are stored in a memory (6) in an object-oriented format, and the graphically representable objects (5, 5*a*) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and the objects thus transformed are combined into an output print data stream (10) and are output, graphically representable objects (5, 5*a*) being stored in the memory (6) in an object-oriented format, to which at least one stored script (5*a*) is assigned, which is executed in the cases defined in the script (5*a*).

53 Claims, 1 Drawing Sheet

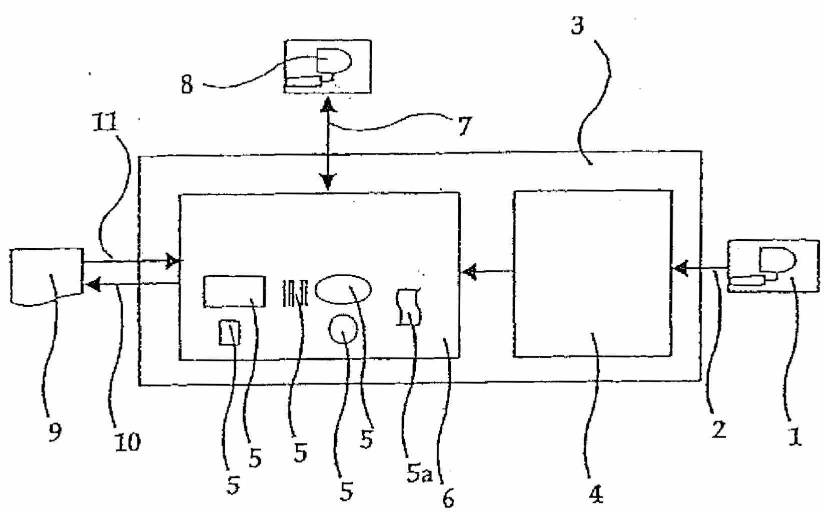
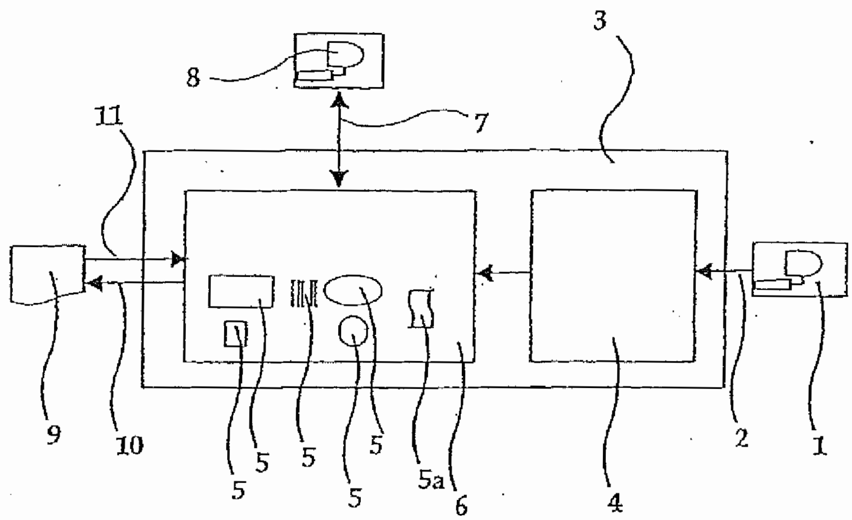


FIG. 1



METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER

The present invention relates to a method and system for the transformation of digital print data streams and corresponding printer and printer server.

Virtually all the output devices which are common nowadays use "page description languages", also called PDL, to produce printed documents. Here, an application program controls a driver for the output device (for example a printer driver). This driver converts information about the graphic objects to be output—for example text or image information—into the respective PDL suitable for the printer used, so that the latter can hereby be controlled directly.

More recent output devices, such as laser printers or digital color printers for example, also offer the possibility of buffering the data streams coming in to control them and, for example, using them as an original form for further incoming print data. This makes it possible to dispense with forms needed for the respective printing, such as letter paper, invoice forms or the like for example, in the individual case. Instead, the application software respectively used merely calls up the form stored once in the printer and combines it with the current print data. In this way, the accumulation of data, for example in networks, can be reduced considerably. However, the result is also organizational advantages: since the forms used no longer have to be kept in reserve by each individual user on his computer, in this way standardized use forms can be achieved, which firstly helps to ensure the often desired standard appearance of a company or an institution and secondly also makes it easier to use current form versions.

However, these aforementioned advantages are normally not used, since the printers used in a company or an institution—with regard to their control—are often not uniform and therefore the use of the functions described above is too complicated, since the appropriate forms either have to be available for each printer model used, which would be very labor-intensive, or only specific printers can be used for specific applications, which is very inflexible.

One possibility of solving this problem is to circumvent the abovedescribed inhomogeneity of the output devices used by employing methods for the conversion of various data stream formats for controlling output devices, which makes it possible for all the computers which produce print data streams to be output to use a standard format for this purpose, by each printer being assigned an interface—be it a dedicated device, be it merely in the form of a software filter—which makes use of such a method and, on the side of the input data stream, uses the format to be used uniformly and, on the side of the output data stream, uses the specific format of the printer to be controlled.

Such a solution is described, for example, by EP 0 109 615 B1, which refers to a method for the conversion of text which is represented in the form of digital data. However, the method taught by this document has considerable disadvantages with regard to the possibilities of current systems from information technology: for example, the method is suitable only for those input print data streams which, in their syntax, follow a format description language whose syntax may be described with the aid of "regular expressions". This is because the method taught in EP 0 109 615 B1 makes use of a status machine, implemented by means of "key status variables", for the recognition and conversion of input control objects recognized in the input print data stream into output control objects. These output control objects are in this case produced directly from the input control objects—specifically in accordance with a fixed assignment—as a function of the respective state representing the key status variables. Such a procedure corresponds to the functioning of the theoretical model of the Moore or Mealy machines, which operate quite efficiently but permit only, the recognition of regular expressions. For these circumstances surrounding information technology at the priority date of EP 0 109 615, such a simple possible transformation may have been sufficient, since—as can already be gathered from claim 1 there—only text had to be converted, apart from format information.

For the current circumstances of PDLs or else other input formats to be recognized where possible, such as HTML or XML, this no longer applies in any way, however. In the meantime, these have been built up in such a complex way with regard to their possibilities that a status machine is no longer in any way adequate for their recognition and conversion.

However, the target format, into which the print data stream is to be transformed, nowadays places high requirements on a transformation: although in principle there would be the possibility here likewise of using the smallest common multiple of the functions of current printing format and in this way of reducing the effort on transformation, this convenience in the design of the transformation process would be brought at great expense in the operation of the method, since in this way the accumulation of data in networks would be increased again, since powerful printer control possibilities which as a rule become more and more specific with regard to the printer type used as the complexity increases, would necessarily have to be dispensed with. Such an increased accumulation of data would, however, again stand in the way of the objective of reducing the data traffic in the network by using PDLs. Thus, at the same time, there is a requirement on the transformation process that the latter produces the preconditions that the target formats can be produced in the most flexible manner possible with all their available printing functions, in order that the traffic on the data transmission lines can thus be minimized.

Furthermore, it is necessary to state that printing systems, even today, still only fulfill a single purpose: namely printing. All the manufacturers of laser printers and digital copying systems have made great efforts in recent years to match the processor powers, storage capacities and additional options (such as memory cards, hard drives, network cards) of these systems to the increasing requirements. However, the manner in which printers and copiers are controlled and programmed has not changed significantly in the last ten years.

Printing systems are still controlled by a page description language (PDL) such as PCL, Postscript or Prescribe. It permits a document and its components to be described adequately. However, the many additional options of modern printing and copying systems available in the meantime cannot be used. The consequence of this is that, even today, the entire printing process is controlled and monitored by a host computer. Its task substantially comprises converting the respective information exactly into the page description language "understood" by the printing system.

It is therefore an object of the present invention to specify a method for the transformation of digital print data streams which is both capable of recognizing more complex page description languages, whose syntax may no longer be described with the aid of simple regular expressions, and also provides the preconditions that the recognized graphic

objects can be transformed into a target format, but also processed further, as flexibly and effectively as possible, that is to say with regard to their description at the highest possible level of abstraction.

According to the invention, this object is achieved by a method for the transformation of digital print data streams, in which an input print data stream is read in, this is analyzed by means of a parser for graphically representable objects and is split up into these graphically representable objects, and the graphically representable objects are stored in a memory in an object-oriented format, and the graphically representable objects stored in the memory in an object-oriented format are transformed into a format for the control of an output device, preferably a printer, and the objects thus transformed are combined into an output print data stream and are output, and which, according to the invention, is characterized in that graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

In this case, as opposed to the use of a status machine, the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used. Instead, such a parser, in terms of its theoretical performance, corresponds to a Turing machine and therefore ensures the theoretically maximum achievable performance for the analysis and splitting up of formal languages.

Furthermore, storing the graphically representable objects—and therefore of course also the scripts, which themselves are certainly also graphically representable objects—in a memory in an object-oriented format achieves the situation where the objects recognized by the parser are then available in this intermediate format which is extremely beneficial for further processing.

The objects are preferably managed here by means of a "display list management", which supports one-page and multi-page documents at as many levels as desired and which can be expanded dynamically by new objects. The individual graphic objects are stored by using their membership of specific—expediently suitably hierarchically organized—classes such as relating to the class of points, ellipses, circles, lines, polygons, rectangles, squares or else to the more complex object types, such as bar codes, more complex texts or freely definable elements such as color profiles or fonts, which permits their effective conversion into an output print data stream, since, through the class of the respective object, there is already implicit information available about its possible transformation into the format of the output print data stream. For example, via an object of the type square, it is already known from the object hierarchy that this is a subclass of the rectangle. If, then, the target format for which an output print data stream is to be produced provides speech constructs relating to the description of rectangles in the page description language, then it is clear, merely on the basis of the position of the square in the object class hierarchy, that this is also a rectangle—albeit with special characteristics—and to this extent the possibilities of the target format with regard to rectangles can also be used for an object in the square class.

In addition to such implicit information—which can be derived from the object class hierarchy—about the individual objects, however, it is also possible to add to the objects explicit information about their possible conversions into specific target formats, it being advantageously possible for this also to be combined with the abovedescribed implicitly provided information, for example by a conversion method into a specific target format being added to a class which is arranged higher in the object class hierarchy, and then automatically also being available to the objects of subordinate, lower-ranking classes by way of inheritance, if a better specified method is not already assigned to said subordinate classes.

In one embodiment of the method according to the invention, the graphically representable objects are combined into super-objects of higher complexity before being stored in the memory.

The super-objects obtained in this way are then stored in the memory in the object-oriented format. In this way, less complex graphic objects can be combined to form more complex graphic super-objects. For example, sequences of lines which in each case join one another at the ends and have been recognized as graphic objects in the input print data stream can be combined to form a graphic polygon super-object. Such a combination offers various advantages, such as easier handling of the super-object stored as a whole as compared with the individual objects, since said super-object can then be treated uniformly by the methods for the super-object class with effect for all the part objects combined in it. It also helps, in certain circumstances, to further minimize the data traffic on the transmission lines used, since an object once combined is subsequently also forwarded in combined form in the output print data stream—if technically supported there—which generally requires a lower data volume to be transmitted than the transmission of the individual objects.

A preferred embodiment of the method according to the present invention is characterized in that a parser is used for the analysis and splitting up into the graphically representable objects, which, in the theoretical model, corresponds to an automatic push-down facility and which is therefore capable of analyzing and splitting up languages with "context-free grammars" particularly effectively.

A further embodiment of the method according to the present invention is characterized in that feedback messages referring to the output print data stream output are read in and are analyzed for error messages which indicate that the output device, preferably the printer, has recognized a transformed graphic object in the output print data stream which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device, are slipped into the output print data stream which is output to the output device.

In this way, it is likewise possible to test whether the driven output device is, for example, capable of recognizing and outputting a bar-code object directly or not. If it is not capable of this and reports this back, then the bar code is simply split up into objects of the next lower hierarchy, for example filled rectangles, and a further try is made with these objects. This is continued until—if necessary until the graphic objects are split up into individual points—the output attempt is successful. The object-oriented data structure with its object hierarchy, chosen for the intermediate format, also proves to be particularly suitable for this procedure. For the further performance of the method, it is preferably noted at which level of the object classes in each case the splitting process was successful for a specific output device, in order then, in the next attempt, already to begin the output process at this level, in order also thus to avoid unnecessary data transfers, but likewise to utilize the maximum level of abstraction of the output device. In this way,

the data volume to be transmitted is reduced to the necessary extent, even with high flexibility.

In an embodiment of the method according to the present invention, at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment, which permits the incorporation of all the devices needed in the widest sense for document processing.

A further preferred embodiment of the method according to the invention is characterized in that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

The script automatically receiving data can preferably also request this data automatically.

It is likewise possible that a script also sends data automatically, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails, it being able in particular also to send the graphic object associated with itself to a receiver.

It can also in turn reassign the data received by it to the graphic object associated with it, and forwards the graphic object associated with itself to a receiver together with the data requested, received and reassigned by itself, or else print out said data.

In relation to the above explanations, it should be noted that the embodiments of the method according to the invention which themselves provide other objects with objects, for example by forwarding them or keeping them ready to receive or for interrogation by a script, for example, are also covered by the term "dynamic object linking" (DOL).

Systems that operate on the method according to the invention, such as printing systems, are capable of sending and receiving e-mails and of printing original print and image data without a printer driver. They are able to store any desired information on hard disks or memory cards and make said data available to all the devices connected in the network and Internet. In other words, they independently undertake demanding tasks in information processing and provision, in order to relieve host computers and personal computers of quite a lot of administrative tasks. In a heterogeneous network and printing environment with laser printing and copying systems from different manufacturers in combination with impact printers and special printing systems, they also make it possible to administer all the connected printing systems with the aid of a single standardized programming language, namely the script language, and therefore reduce the effort on administration to a minimum. At this point, it should be mentioned that these systems operating by the method according to the invention are also designated JScribe (registered trademark) systems and, accordingly, the method according to the invention is also designated JScribe (registered trademark).

When JScribe (registered trademark) is used, developers and system houses will therefore be in a position to provide objects and functions which are stored in resident form in the printing system and permit and control desired individual operating sequences. These objects and functions can use any functionality provided by the JScribe (registered trademark) basic technology, including extremely demanding commands for the job or page processing and for the complete control of the print data and emulations. The method according to the invention preferably also enables access to internal printer functions and status information (page counter, network components, file system and so on), for example via a script.

The method according to the invention is preferably characterized in that graphically representable objects are stored in the memory in an object-oriented format, to which at least one stored script is assigned, which is executed in the case of the output of the object defined in the script. In this way, for example, it is possible to execute such scripts, for example Visual Basic Scripts, Java Scripts or else "stream code" in an event-oriented manner, for example in the case where a form object is printed out, likewise "ON-PRINT" by which means, for example, to execute such functions as the printing of copies of the same form with the same net data but on different paper from different trays. In particular in interaction with those embodiments of the method according to the invention which control external devices, such as folding or enveloping machines or else stapling machines, this is particularly advantageous.

However, it may also be the case that at least one case relating to the execution of the script is defined in the respective script and occurs automatically, preferably without further influence from outside.

For example, the automatically occurring case, defined at least in the respective script, relating to the execution of the script can be configured as a timer, that is to say as a case which occurs automatically as a result of the expiry of a time, this timer preferably operating cyclically, that is to say starting itself again upon expiry.

Automatic scripts can therefore intrinsically become active and, for example, load the daily newspaper, where possible itself assembled from different sources, from the Internet, assign the found, loaded and analyzed information to a stored object and then print this object, completely without the participation of a PC or other host computer to which the printer would be connected.

For example, the simple download of JScribe (registered trademark) sequences (scripts with appropriately associated objects) can, for example, arrange for the printer automatically to fetch information about current share prices, to format it and to print it out. Image information, text documents, web pages, XML documents and any other desired print data can be analyzed while dispensing with any preparation by the PC (for example by a printer driver), modified if necessary and printed out in optimum quality. Since JScribe (registered trademark) can also be employed simultaneously as a server version for computer systems, printing systems are for the first time made capable of accessing stocks of data on host systems (for example SQL databases) interactively during the printing operation.

The language used for the scripts according to the present invention is preferably Java Script. Java Script, as a world-established standard for the script-controlled, intelligent programming of web pages, has triggered in the Internet an avalanche of innovative and functional solutions which have contributed decisively to ringing in the age of eBusiness and eCommerce. This intelligent technology, which has so decisively marked the worldwide, rapid development of the Internet, is therefore now also available for printing systems for the first time and here preferably forms the basic technology for script applications in the area of the present invention, and consequently print and document management, which is certainly uniquely and, as compared with established solutions, considerably more cost-effective.

With JScribe in conjunction with Java Script, an innovative technology is therefore provided which allows any

corresponding print system operated in accordance with the method according to the invention to be programmed just as simply as an Internet homepage. The communications possibilities already described, together with the logically modular object-oriented construction of JScribe and the Java-Script-typical expansion possibilities ideally supporting JScribe permit within the shortest possible time the construction of complex output management systems for an extremely wide range of applications.

A further preferred embodiment of the method according to the present invention is characterized in that the graphically representable objects stored in the memory in an object-oriented format, preferably also script objects (for example Java Script objects), preferably before they are output in the output print data stream, are kept ready by an application interface to be read out, to be changed, to be deleted or to have new objects appended.

According to the prior art, hitherto the page descriptions necessary for the storage of forms in the output devices had to be created laboriously by hand, that is to say programmed in the respective page description language—time-consuming and expensive work which can be carried out only by a few programmers qualified to do this. The same also applies to changes in the stored data.

The object-oriented intermediate format now makes it possible for the stored graphically representable objects to be kept ready to be read out, to be changed, to be deleted or for new objects to be appended, in a technically elegant manner via an application interface, by assigning the methods required for this to the respective objects in accordance with their class hierarchy. This means that the objects stored in the memory can, for example, be displayed on a screen and modified as desired. Here, too, deleting existing objects and appending new objects are also possible.

By means of binding suitable application software—also called FormMaker—it is therefore made possible in particular for each EDP user to modify existing forms and to create new forms entirely without any programming knowledge, which likewise applies to scripts.

Given suitable selection of the application interface and processing methods correspondingly available to a sufficient extent for the object classes used, a graphic core system with a functional interface is thus made available, which can be used by applications for graphic user interfaces, such as those based on the Windows operating system, to display the object data as a standard document on the screen and to modify it with different processing tools.

The application interface also preferably permits script objects, preferably Java Script objects themselves, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations being automatically transformed, if required, into script objects, preferably Java Script objects. It thus provides a complete graphic development environment for computers, preferably computers operating under the Windows operating system, which permits the printing and copying systems to be programmed without Java Script knowledge.

In addition, already existing development tools which are based on Java can likewise be used for the development of individual JScribe (registered trademark) applications.

The use of "FormMaker" application software permits the design of "intelligent" electronic forms, which are transformed into logical documents with the aid of JScribe (registered trademark). These in turn can be made available in systems connected to the network and output at any desired location by any desired printing systems, preferably laser printing systems and digital copying systems, sent as e-mail or else transferred to archiving systems.

The present method according to the invention can also be present implemented on a system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, the data processing unit being programmed in such a way that it operates in accordance with an embodiment of the method according to the invention.

In this case, the system preferably also has an operating station with display means and input means, which makes it possible for the graphically representable objects stored in the memory of the data processing unit in an object-oriented format, preferably also script objects, to be read out via the application interface, to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream.

In addition, the system according to the invention can moreover permit respectively stored objects, preferably even script objects themselves, such as Java Script objects, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations being transformed automatically, if required, into Java Script objects.

The system according to the invention can also be integrated into a printer or else a printer server.

JScribe (registered trademark) can therefore not only be employed directly on printers and digital copying system but can also be implemented on PC server platforms.

For installation purposes on printing systems, the JScribe script sequences can, for example, be incorporated into a Prescribe (registered trademark) data stream. The printing system receiving this data, for example the appropriate laser printer or digital copier, will read in and compile the program code.

This permits the configuration of networks with hardware units which are small but equipped with high functionality, which have a common interface and permit access relating to archiving documents, to distributed printing (cluster printing) and security printing and much more.

The abovedescribed embodiments of the method according to the present invention can of course in each case also be implemented as a computer program product which has a computer-readable medium with computer program code means or as a computer program on an electronic carrier signal and in which, in each case after the computer program has been loaded, the computer is caused by the program to carry out the method according to the invention described here.

In the following text, an exemplary embodiment, not to be understood as restrictive, will be discussed by using the drawing, in which:

FIG. 1 shows the sequence of an embodiment of the method according to the invention using a schematic representation.

FIG. 1 shows the sequence of an embodiment of the method according to the invention using a schematic representation. From a computer 1, an input print data stream 2 is sent to a device 3—for example a computer such as a PC or else an intelligent output device such as an intelligent printer—which operates in accordance with the method according to the present invention. There, the input print data stream 2 is analyzed and split up by a parser 4. The graphic objects **5**, **5a** recognized as the product of this splitting are stored in a memory 6 in an object-oriented format; this is after they have possibly been combined to form super-objects. The objects 5 stored in the memory 6, preferably script objects 5a, are kept ready to be read out via

an application interface 7, to be changed, to be deleted or for new objects to be appended. In this way, the objects 5, 5*a* stored in the memory 6 can, for example, be displayed on a screen 9 and modified as desired. Deleting existing objects and appending new objects is also possible here. If suitable application software is used, it is thus possible for any user to modify existing forms easily and without programming knowledge or to create new forms easily and without programming knowledge or to create new forms. The graphically representable objects 5, 5*a* stored in the memory 6 in an object-oriented format are transformed into a format for the control of an output device, preferably a printer 9, in order to be output, and the objects 5, 5*a* thus transformed are combined into an output print data stream 10 and output. Feedback messages 11 concerning the output print data stream 10 output are read in and analyzed for error messages which indicate that the printer 10 has detected a graphic object 5, 5*a* in the output print data stream 10 which cannot be output or processed by said printer. This graphic object 5, 5*a* is then split up into part objects of lower complexity, and the part objects obtained in this way, in the format for the control of the printer 9, are slipped into the output print data stream 10 which is output to the printer 9.

What is claimed is:

1. A method for the transformation of digital print data streams, in which

(i) an input print data stream (2) is read in,

(ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and

(iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, and

(iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (10) and are output,

characterized in that graphically representable objects (5, 5*a*) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

2. The method as claimed in claim 1, characterized in that the graphically representable objects (5, 5*a*) are combined into super-objects of higher complexity before being stored in the memory (6).

3. The method as claimed in claim 1, characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

4. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5*a*) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

5. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5*a*) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

6. The method as claimed in claim 5, characterized in that the script (5*a*) automatically receiving data also requests this data automatically.

7. The method as claimed in claim 5, characterized in that the script (5*a*) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

8. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5*a*) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

9. The method as claimed in claim 8, characterized in that the script (5*a*) sends the graphic object (5) associated with itself to receiver.

10. The method as claimed in claim 9, characterized in that the script (5*a*) in turn reassigns the data received by it to the graphic object (5) associated with it, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

11. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5*a*) which is executed in the case of the output of the object (5) defined in the script (5*a*).

12. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5*a*), at least one case relating to the execution of the script (5*a*) being defined in the respective script (5*a*), and occurring automatically, preferably without further influence from outside.

13. The method as claimed in claim 12, characterized in that the automatically occurring case, defined at least in the respective script (5*a*), relating to the execution of the script (5*a*) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

14. The method as claimed in claim 13, characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

15. The method as claimed in claim 1, characterized in that Java Script is used as a formal language for the scripts.

16. The method as claimed in claim 1, characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5*a*), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

17. A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates in accordance with the method as claimed claim 1.

18. The system for the transformation of digital print data streams as claimed in claim 17, the system also has an operating station with display means (8) and input means, which makes it possible for the graphically representable objects (5) stored in the memory (6) of the data processing

unit in an object-oriented format, preferably also script objects (5a), to be read out via the application interface (7), to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream (10).

19. The system for the transformation of digital print data streams as claimed in claim 17, wherein the data processing unit, permits respectively stored objects, preferably also Java Script objects (5a) themselves, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations if necessary being transformed automatically into Java Script objects (5a).

20. A printer, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.

21. A printer server, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.

22. A computer-readable medium having stored thereon instructions to cause a processor to execute a method, the method comprising:

(i) an input print data stream (2) is read in,

(ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and

(iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format,

(iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (10) and are output,

characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

23. The computer-readable medium as claimed in claim 22, the method characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

24. The computer-readable medium as claimed in claim 22, the method characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

25. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

26. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

27. The computer-readable medium as claimed in claim 26, the method characterized in that the script (5a) automatically receiving data also requests this data automatically.

28. The computer-readable medium as claimed in claim 26, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigend by itself.

29. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

30. The computer-readable medium as claimed in claim 29, the method characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

31. The computer-readable medium as claimed in claim 30, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and forwards the graphic object (5) associated with itself to a receiver together with the data requested, received and reassigned by itself.

32. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

33. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

34. The computer-readable medium as claimed in claim 33, the method characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

35. The computer-readable medium as claimed in claim 34, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

36. The computer-readable medium as claimed in claim 22, the method characterized in that Java Script is used as a formal language for the scripts.

37. The computer-readable medium as claimed in claim 22, the method characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

38. A computer data signal embodied in a carrier wave and representing sequences of instructions which, when executed by a processor, cause the processor to perform a method, the method comprising:

(i) an input print data stream (2) is read in,

(ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and

(iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format,

(iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (10) and are output,

characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

39. The computer data signal as claimed in claim 38, the method characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

40. The computer data signal as claimed in claim 38, the method characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer, this graphic object is then split up into part objects of lower complexity, and the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

41. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

42. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

43. The computer data signal as claimed in claim 42, the method characterized in that the script (5a) automatically receiving data also requests this data automatically.

44. The computer data signal as claimed in claim 42, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

45. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

46. The computer data signal as claimed in claim 45, the method characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

47. The computer data signal as claimed in claim 46, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and forwards the graphic object (5) associated with itself to a receiver together with the data requested, received and reassigned by itself.

48. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

49. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

50. The computer data signal as claimed in claim 49, the method characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

51. The computer data signal as claimed in claim 50, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

52. The computer data signal as claimed in claim 38, the method characterized in that Java Script is used as a formal language for the scripts.

53. The computer data signal as claimed in claim 38, the method characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/275,784	02/03/2004	6684789	60428.00001	5658

30256 7590 01/15/2004

SQUIRE, SANDERS & DEMPSEY L.L.P
600 HANSEN WAY
PALO ALTO, CA 94304-1043

DATES ENTERED
No Action

JAN 2 0 2004

CALENDARED
BY
ATTORNEY
SQUIRE, SANDERS & DEMPSEY

ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment is 0 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) system (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (703) 305-1383. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at (703) 305-8283.

APPLICANT(S):

Thomas Krautter, Stuttgart, GERMANY;

Date Mailed: November 3, 2003 | By: AW/say	PTO DATE STAMP:
Serial No.: 10/275,784	Docket No.: 60428.00001
Applicant: Thomas Krautter	
Title: METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER	

The following has been received in the U.S. Patent Office on the date stamped hereon:

- ☐ Patent Application ___ Pages ___ Claims
- ☐ Drawings Formal ___ Sheets
- ☐ Oath/Declaration/Power of Attorney
- ☐ Assignment & Recordation Cover Sheet
- ☐ Verified Statement Claiming Small Entity Status
- ☐ Supplemental Info. Disclosure Statement & PTO-1449 / Refs ___
- ☐ Provisional Application _____ Pages
- ☒ Authorization to Charge Deposit Account No. 05-0150 ($1330 Issue Fee & $300 Publication Fee)
- ☒ Certificate(s) of First Class Mailing
- ☐ Amendment/Response
- ☐ Petition for Extension of Time
- ☒ Transmittal Form
- ☐ Notice of Appeal
- ☐ Fee Transmittal for FY 2002 (in duplicate)
- ☒ PTOL-85 Issue Fee Transmittal (in duplicate)
- ☐ Copy of PTO-1533, Notice to File Missing Parts
- ☐ Check No. for $

U.S DEPARTMENT OF COMMERCE
PATENT/TRADEMARK OFFICE
WASHINGTON, D.C. 20231

PALO ALTO CA

U.S. POSTAGE 00.23
H METER 711076

Squire, Sanders & Dempsey L.L.P.
600 Hansen Way, Suite 100
Palo Alto, CA 94304-1043

PTO/SB/21 (05-03)
Approved for use through 04/30/2003. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	10/275,784
Filing Date	November 7, 2002
First Named Inventor	Thomas Krautter
Art Unit	2854
Examiner Name	Chau, Minh H.
Attorney Docket Number	60428.00001

Total Number of Pages in This Submission: 3

ENCLOSURES *(check all that apply)*

- ☐ Fee Transmittal Form
- ☐ Request for Corrected Filing Receipt
- ☐ Amendment / Response
  - ☐ After Final
  - ☐ With RCE
- ☐ Extension of Time Request
- ☐ Return Postcard
- ☐ IDS and Form 1449
- ☐ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
- ☐ Declaration/Oath
- ☐ Assignment and Recordation Cover Sheet *(for an Application)*
- ☐ Formal Drawings ____ Sheets
- ☐ Licensing-related Papers
- ☐ Petition
- ☒ PTOL-85 Issue Fee Transmittal (in duplicate)
- ☐ Letter to the Official Draftsperson (Submission of Formal Drawings)
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) _____
- ☐ After Allowance Communication to Group
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- ☐ Proprietary Information
- ☒ Authorization to Charge Deposit Account No. 05-0150 ($1330 Issue Fee & $300 Publication Fee)
- ☐ Other Enclosure(s) *(please identify below)*:

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm *or* Individual name	Aaron Wininger, Reg. No. 45,229 Squire, Sanders & Dempsey L.L.P. 600 Hansen Way Palo Alto, CA 94304-1043
Signature	
Date	November 3, 2003

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Typed or printed name	Aaron Wininger		
Signature		Date	November 3, 2003

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (703) 746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 4 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Legibly mark-up with any corrections or use Block 1)

30256 7590 10/07/2003

SQUIRE, SANDERS & DEMPSEY L.L.P
600 HANSEN WAY
PALO ALTO, CA 94304-1043

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO, on the date indicated below.

Aaron Wininger (Depositor's name)
[signature] (Signature)
November 3, 2003 (Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/275,784	11/07/2002	Thomas Krautter	60428.00001	5658

TITLE OF INVENTION: METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1330	$300	$1630	01/07/2004

EXAMINER	ART UNIT	CLASS-SUBCLASS
CHAU, MINH H	2854	101-484000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 Squire, Sanders & Dempsey L.L.P.
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the USPTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: CCP Systems AG
(B) RESIDENCE: (CITY and STATE OR COUNTRY): Stuttgart, Germany

Please check the appropriate assignee category or categories (will not be printed on the patent); ☐ individual XX corporation or other private group entity ☐ government

4a. The following fee(s) are enclosed:

X Issue Fee
X Publication Fee
☐ Advance Order - # of Copies ________

4b. Payment of Fee(s):

☐ A check in the amount of the fee(s) is enclosed.
☐ Payment by credit card. Form PTO-2038 is attached.
X The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 05-0150 (enclose an extra copy of this form).

Director for Patents is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.

(Authorized Signature) [signature] (Date) 11/3/03
Aaron Wininger, Reg. No. 45,229

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S.

COPY

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (703) 746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 4 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Legibly mark-up with any corrections or use Block 1)

30256 7590 10/07/2003

SQUIRE, SANDERS & DEMPSEY L.L.P
600 HANSEN WAY
PALO ALTO, CA 94304-1043

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO, on the date indicated below.

Aaron Wininger (Depositor's name)
[signature] (Signature)
November 3, 2003 (Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/275,784	11/07/2002	Thomas Krautter	60428.00001	5658

TITLE OF INVENTION: METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1330	$300	$1630	01/07/2004

EXAMINER	ART UNIT	CLASS-SUBCLASS
CHAU, MINH H	2854	101-484000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. Use of a Customer Number is required.

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 Squire, Sanders & Dempsey L.L.P.
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the USPTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: CCP Systems AG
(B) RESIDENCE: (CITY and STATE OR COUNTRY): Stuttgart, Germany

Please check the appropriate assignee category or categories (will not be printed on the patent): ☐ individual XX corporation or other private group entity ☐ government

4a. The following fee(s) are enclosed:
- X Issue Fee
- X Publication Fee
- ☐ Advance Order - # of Copies ____

4b. Payment of Fee(s):
- ☐ A check in the amount of the fee(s) is enclosed.
- ☐ Payment by credit card. Form PTO-2038 is attached.
- X The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 05-0150 (enclose an extra copy of this form).

Director for Patents is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.

(Authorized Signature) [signature] (Date) 11/3/03
Aaron Wininger, Reg. No. 45,229

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

DATES ENTERED

NOTICE ~~OF ALLOWANCE AND~~ FEE(S) DUE

Issue Fee Due
1/7/2004

30256 7590 10/07/2003

SQUIRE, SANDERS & DEMPSEY L.L.P
600 HANSEN WAY
PALO ALTO, CA 94304-1043

OCT 10 2003

CALENDARED

BY ~~[illegible]~~
ATTORNEY
SQUIRE, SANDERS & DEMPSEY

EXAMINER	
CHAU, MINH H	
ART UNIT	PAPER NUMBER
2854	

DATE MAILED: 10/07/2003

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/275,784	11/07/2002	Thomas Krautter	60428.00001	5658

TITLE OF INVENTION: METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1330	$300	$1630	01/07/2004

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE REFLECTS A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE APPLIED IN THIS APPLICATION. THE PTOL-85B (OR AN EQUIVALENT) MUST BE RETURNED WITHIN THIS PERIOD EVEN IF NO FEE IS DUE OR THE APPLICATION WILL BE REGARDED AS ABANDONED.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status is changed, pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above and notify the United States Patent and Trademark Office of the change in status, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check the box below and enclose the PUBLICATION FEE and 1/2 the ISSUE FEE shown above.

☐ Applicant claims SMALL ENTITY status. See 37 CFR 1.27.

II. PART B - FEE(S) TRANSMITTAL should be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). Even if the fee(s) have already been paid, Part B - Fee(s) Transmittal should be completed and returned. If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail — Mail Stop ISSUE FEE, Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450

or Fax (703) 746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 4 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Legibly mark-up with any corrections or use Block 1)

30256 7590 10/07/2003

SQUIRE, SANDERS & DEMPSEY L.L.P
600 HANSEN WAY
PALO ALTO, CA 94304-1043

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO, on the date indicated below.

(Depositor's name)

(Signature)

(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/275,784	11/07/2002	Thomas Krautter	60428.00001	5658

TITLE OF INVENTION: METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1330	$300	$1630	01/07/2004

EXAMINER	ART UNIT	CLASS-SUBCLASS
CHAU, MINH H	2854	101-484000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 ____________

2 ____________

3 ____________

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the USPTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent); ☐ individual ☐ corporation or other private group entity ☐ government

4a. The following fee(s) are enclosed:

☐ Issue Fee

☐ Publication Fee

☐ Advance Order - # of Copies ____________

4b. Payment of Fee(s):

☐ A check in the amount of the fee(s) is enclosed.

☐ Payment by credit card. Form PTO-2038 is attached.

☐ The Director is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number ____________ (enclose an extra copy of this form).

Director for Patents is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.

(Authorized Signature) (Date)

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Alexandria, Virginia

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/275,784	11/07/2002	Thomas Krautter	60428.00001	5658

30256 7590 10/07/2003

SQUIRE, SANDERS & DEMPSEY L.L.P
600 HANSEN WAY
PALO ALTO, CA 94304-1043

EXAMINER	
CHAU, MINH H	
ART UNIT	**PAPER NUMBER**
2854	

DATE MAILED: 10/07/2003

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 0 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 0 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) system (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (703) 305-1383. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at (703) 305-8283.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/275,784	11/07/2002	Thomas Krautter	60428.00001	5658

30256 7590 10/07/2003

SQUIRE, SANDERS & DEMPSEY L.L.P
600 HANSEN WAY
PALO ALTO, CA 94304-1043

EXAMINER	
CHAU, MINH H	
ART UNIT	**PAPER NUMBER**
2854	

DATE MAILED: 10/07/2003

Notice of Fee Increase on October 1, 2003

If a reply to a "Notice of Allowance and Fee(s) Due" is filed in the Office on or after October 1, 2003, then the amount due will be higher than that set forth in the "Notice of Allowance and Fee(s) Due" since there will be an increase in fees effective on October 1, 2003. See Revision of Patent Fees for Fiscal Year 2004; Final Rule, 68 Fed. Reg. 41532, 41533, 41534 (July 14, 2003).

The current fee schedule is accessible from (http://www.uspto.gov/main/howtofees.htm).

If the fee paid is the amount shown on the "Notice of Allowance and Fee(s) Due" but not the correct amount in view of the fee increase, a "Notice of Pay Balance of Issue Fee" will be mailed to applicant. In order to avoid processing delays associated with mailing of a "Notice of Pay Balance of Issue Fee," if the response to the Notice of Allowance is to be filed on or after October 1, 2003 (or mailed with a certificate of mailing on or after October 1, 2003), the issue fee paid should be the fee that is required at the time the fee is paid. If the issue fee was previously paid, and the response to the "Notice of Allowance and Fee(s) Due" includes a request to apply a previously-paid issue fee to the issue fee now due, then the difference between the issue fee amount at the time the response is filed and the previously-paid issue fee should be paid. See Manual of Patent Examining Procedure, Section 1308.01 (Eighth Edition, August 2001).

Effective October 1, 2003, 37 CFR 1.18 is amended by revising paragraphs (a) through (c) to read as set forth below.

Section 1.18 Patent post allowance (including issue) fees.

(a) Issue fee for issuing each original or reissue patent, except a design or plant patent:
By a small entity (Sec. 1.27(a))...................... $665.00
By other than a small entity......................... $1,330.00

(b) Issue fee for issuing a design patent:
By a small entity (Sec. 1.27(a))...................... $240.00
By other than a small entity........................... $480.00

(c) Issue fee for issuing a plant patent:
By a small entity (Sec. 1.27(a))...................... $320.00
By other than a small entity........................... $640.00

Questions relating to issue and publication fee payments should be directed to the Customer Service Center of th Office of Patent Publication at (703) 305-8283.

Notice of Allowability	Application No. 10/275,784	Applicant(s) KRAUTTER, THOMAS
	Examiner Minh H Chau	Art Unit 2854

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *the Application filed on 07November 2002*.
2. ☒ The allowed claim(s) is/are *1-53*.
3. ☒ The drawings filed on *07 November 2002* are accepted by the Examiner.
4. ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☒ All b) ☐ Some* c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. ____ .
   3. ☒ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: ____.
5. ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application).
   (a) ☐ The translation of the foreign language provisional application has been received.
6. ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application. **THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.**

7. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

8. ☐ CORRECTED DRAWINGS must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
   1) ☐ hereto or 2) ☐ to Paper No. ____.
   (b) ☐ including changes required by the proposed drawing correction filed ____, which has been approved by the Examiner.
   (c) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No. ____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet.

9. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1☒ Notice of References Cited (PTO-892)
3☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
5☒ Information Disclosure Statements (PTO-1449), Paper No. *2*.
7☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
2☐ Notice of Informal Patent Application (PTO-152)
4☐ Interview Summary (PTO-413), Paper No.____ .
6☐ Examiner's Amendment/Comment
8☒ Examiner's Statement of Reasons for Allowance
9☐ Other

Notice of References Cited	Application/Control No. 10/275,784	Applicant(s)/Patent Under Reexamination KRAUTTER, THOMAS	
	Examiner Minh H Chau	Art Unit 2854	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-5,566,278	10-1996	Patel et al.	358/1.15
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

PCT/PTO 07 NOV 2

PTO/SB/08A (10-01)
Approved ... through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

#2

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	~~Unknown~~ 10/275,784
Filing Date	November 7, 2002
First Named Inventor	Thomas Krautter
Group Art Unit	~~Unknown~~ 2854
Examiner Name	~~Unknown~~ MINH CHAU
Attorney Docket Number	60428.00001

U.S. PATENT DOCUMENTS

Examiner Initials *	Cite No.[1]	Document Number: Number - Kind Code[2] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
MC		US-5,216,754	06-01-1993	Kitty Sathi, et al.	—
MC		US- 6,006,013	12-21-1999	David E. Rumph, et al.	—
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			
		US-			

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document: Country Code[3] - Number[4] - Kind Code[5]	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T[6]
MC		EP - 0 109 615 - B1	06-01-1994	International Business Machines Corporation	—	—
MC		EP - 0 964 339 - A2	12-15-1999	NEC Corporation	—	—
MC		EP - 1 061 456 - A2	12-20-2000	NEC Corporation	—	—
MC		GB - 2 357 348 - A	06-20-2001	International Business Machines Corporation	—	—
MC		WO - 00/17748	03-30-2000	Netcreate Systems, Inc.	—	—
MC		WO - 00/55720	09-21-2000	Prout AG	—	—

Examiner Signature	MINH CHAU	Date Considered	09-24-03

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

[1] Applicant's unique citation designation number (optional). [2] See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. [3] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). [4] For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. [5] Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST. 16 if possible. [6] Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

REASONS FOR ALLOWANCE

1. The following is an examiner's statement of reasons for allowance:

Claims 1-53 have been indicated for allowance because the prior art fails to teach the entire combination of a method for the transformation of digital print data stream including a steps of reading an input print data stream, analyzing the input data stream by means of a parser and splitting the input data into the graphically representable objects, storing the graphically representable objects in a memory in an object-oriented format, transforming the object-oriented format into a format for controlling a printer, the object-oriented format stored in the memory including at least one stored script is assigned, which is executed in the case defined in the script.

2. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance.".

3. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Minh H Chau whose telephone number is (703) 305-0298. The examiner can normally be reached on M - TH.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Andrew H Hirshfeld can be reached on (703) 305-6619. The fax phone number for the organization where this application or proceeding is assigned is (703) 872-9306.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 308-0956.

REVISED AMENDMENT PRACTICE: 37 CFR 1.121 CHANGED
COMPLIANCE IS MANDATORY - Effective Date: July 30, 2003

All amendments filed on or after the effective date noted above must comply with revised 37 CFR 1.121. See Final Rule: **Changes To Implement Electronic Maintenance of Official Patent Application Records** (68 *Fed. Reg.* 38611 (June 30, 2003), posted on the Office's website at: http://www.uspto.gov/web/patents/ifw/ with related information. The amendment practice set forth in revised 37 CFR 1.121, and described below, replaces the voluntary revised amendment format available to applicants since February 2003. **NOTE: STRICT COMPLIANCE WITH THE REVISED 37 CFR 1.121 IS REQUIRED AS OF THE EFFECTIVE DATE** (July **30, 2003).** The Office will notify applicants of amendments that are not accepted because they do not comply with revised 37 CFR 1.121 via a Notice of Non-Compliant Amendment. See MPEP 714.03 (Rev. 1, Feb. 2003). The non-compliant section(s) will have to be corrected and the entire corrected section(s) resubmitted within a set period.

<u>Bold underlined italic font has been used below to highlight the major differences between the revised 37 CFR 1.121 and the voluntary revised amendment format that applicants could use since February, 2003.</u>
Note: The amendment practice for **reissues** and **reexamination proceedings**, except for drawings, has not changed.

<u>REVISED AMENDMENT PRACTICE</u>

<u>I. Begin each section of an amendment document on a separate sheet:</u>
Each section of an amendment document (*e.g.*, Specification Amendments, Claim Amendments, Drawing Amendments, and Remarks) must begin on a separate sheet. Starting each separate section on a new page will facilitate the process of separately indexing and scanning each section of an amendment document for placement in an image file wrapper.

<u>II. Two versions of amended part(s) no longer required:</u>
37 CFR 1.121 has been revised to <u>no longer require</u> two versions (a clean version and a marked up version) of each replacement paragraph or section, or amended claim. Note, however, the requirements for a clean version and a marked up version for <u>substitute specifications</u> under 37 CFR 1.125 have been retained.

A) **<u>Amendments to the claims:</u>**
Each amendment document that includes a change to an existing claim, cancellation of a claim or submission of a new claim, **must include a complete listing** of all claims in the application. After each claim number in the listing, the status must be indicated in a parenthetical expression, and **the text of each pending claim** (with markings to show <u>current</u> changes) must be presented. The claims in the listing will replace all prior claims in the application.

(1) The current status of all of the claims in the application, including any previously canceled, not entered or withdrawn claims, must be given in a parenthetical expression following the claim number using only one of the following seven status identifiers: (original), (currently amended), (canceled), (withdrawn), (new), ***<u>(previously presented) and (not entered).</u>*** The text of all pending claims, ***<u>including withdrawn claims,</u>*** must be submitted each time any claim is amended. Canceled ***<u>and not entered</u>*** claims must be indicated by only the claim number and status, without presenting the text of the claims.

(2) The text of all claims <u>being currently amended</u> must be presented in the claim listing with markings to indicate the changes that have been made relative to the immediate prior version. The changes in any amended claim must be shown by underlining (for added matter) or strikethrough (for deleted matter) with 2 exceptions: (1) for ***<u>deletion of five characters or fewer, double brackets may be used (e.g., [[eroor]]); and (2) if strikethrough cannot be easily perceived (e.g., deletion of the number "4" or certain punctuation marks), double brackets must be used (e.g., [[4]]). As an alternative to using double brackets, however, extra portions of text may be included before and after text being deleted, all in strikethrough, followed by including and underlining the extra text with the desired change</u>*** (*e.g.*, ~~*number 4 as*~~ *<u>number 14 as</u>*). An accompanying clean version is not required and should not be presented. Only claims of the status "currently amended," and "withdrawn" that are being amended, may include markings.

(3) The text of pending claims not being currently amended, ***including withdrawn claims***, must be presented in

The United States Patent and Trademark Office has changed certain mailing addresses!

Effective May 1, 2003

Use the address **provided in this flyer after May 1, 2003** for **any correspondence** with the United States Patent and Trademark Office (USPTO) in patent-related matters to **organizations reporting to the Commissioner for Patents.**

DO NOT USE the Washington DC 20231 and P.O. Box 2327 Arlington, VA 22202 addresses after May 1, 2003 for **any correspondence** with the USPTO even if these old addresses are indicated in the accompanying Office action or Notice or in any other action, notice, material, form, instruction or *other* information.

Correspondence in patent-related matters to **organizations reporting to the Commissioner for Patents** must <u>now</u> be addressed to:

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Special Mail Stop designations to replace Special Box designations

Also effective May 1, 2003, the **USPTO is changing the special Box designations for Patents and Trademarks to corresponding Mail Stop designations** (*e.g.*, "Box 4" will now be "Mail Stop 4").

For further information, see *Correspondence with the United States Patent and Trademark Office*, 68 *Fed. Reg. 14332* (March 25, 2003). A copy of the *Federal Register* notice is available on the USPTO's web site at http://www.uspto.gov/web/menu/current.html#register

A listing of specific USPTO mailing addresses (See Patents – specific) will be available on the USPTO's web site on April 15, 2003 at http://www.uspto.gov/main/contacts.htm

Persons filing correspondence with the Office should check the rules of practice, the <u>Official Gazette</u>, or the Office's Internet Web site (www.uspto.gov) to determine the appropriate address and Mail Stop Designation (if applicable) for all correspondence being delivered to the USPTO via the United States Postal Service (USPS).

Questions regarding the content of this flyer should be directed to the Inventor Assistance Center at (703) 308-4357 or toll-free at 1-800-786-9199.

ALLOWED CLAIMS
For U.S. Patent Application No. 10/275,784
Filed November 7, 2002
Entitled: Method and System for the Transformation
of Digital Print Data Streams and Corresponding Printer and Printer Server
Inventor: Thomas Krautter

1. A method for the transformation of digital print data streams, in which

   (i) an input print data stream (2) is read in,

   (ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and

   (iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format, and

   (iv) the graphically representable objects (5) stored in the memory (6) in an objected-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and

   (v) the objects thus transformed are combined into an output print data stream (10) and are output;

   characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an objected-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

2. The method as claimed in claim 1, characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

3. The method as claimed in claim 1, characterized in that

   feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer,

   this graphic object is then split up into part objects of lower complexity, and

the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

4. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

5. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

6. The method as claimed in claim 5, characterized in that the script (5a) automatically receiving data also requests this data automatically.

7. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

8. The method as claimed in claim 7, characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

9. The method as claimed in claim 8, characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and forwards the graphic object (5) associated with itself to a receiver together with the data requested, received and reassigned by itself.

10. The method as claimed in claim 5, characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

11. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

12. The method as claimed in claim 1, characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

13. The method as claimed in claim 12, characterized in that the automatically occurring case, defined at least in the respective script 5(a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

14. The method as claimed in claim 13, characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

15. The method as claimed in claim 1, characterized in that Java Script is used as a formal language for the scripts.

16. The method as claimed in claim 1, characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print

data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

17. A system for the transformation of digital print data streams comprising at least one data processing unit having at least one memory and at least one communications interface, characterized in that the data processing unit is programmed in such a way that it operates in accordance with the method as claimed in claim 1.

18. The system for the transformation of digital print data streams as claimed in claim 17, the system also has an operating station with display means (8) and input means, which makes it possible for the graphically representable objects (5) stored in the memory (6) of the data processing unit in an object-oriented format, preferably also script objects (5a), to be read out via the application interface (7), to be changed, to be deleted or to be appended, preferably before they are output in the output print data stream (10).

19. The system for the transformation of digital print data streams as claimed in claim 17, wherein the data processing unit permits respectively stored objects, preferably also Java Script objects (5a) themselves, to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations if necessary being transformed automatically into Java Script objects (5a).

20. A printer, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.

21. A printer server, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17.

22. A computer-readable medium having stored thereon instructions to cause a processor to execute a method, the method comprising:

(i) an input print data stream (2) is read in,

(ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and

(iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format,

(iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (10) and are output,

characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

23. The computer-readable medium as claimed in claim 22, the method characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

24. The computer-readable medium as claimed in claim 22, the method characterized in that

feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer,

this graphic object is then split up into part objects of lower complexity, and

the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

25. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the

object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

26. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

27. The computer-readable medium as claimed in claim 26, the method characterized in that the script (5a) automatically receiving data also requests this data automatically.

28. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

29. The computer-readable medium as claimed in claim 28, the method characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

30. The computer-readable medium as claimed in claim 29, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with it, and forwards the graphic object (5) associated with itself to a receiver together with the data requested, received and reassigned by itself.

31. The computer-readable medium as claimed in claim 26, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5)

associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

32. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

33. The computer-readable medium as claimed in claim 22, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

34. The computer-readable medium as claimed in claim 33, the method characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

35. The computer-readable medium as claimed in claim 34, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

36. The computer-readable medium as claimed in claim 22, the method characterized in that Java Script is used as a formal language for the scripts.

37. The computer-readable medium as claimed in claim 22, the method characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7)

38. A computer data signal embodied in a carrier wave and representing sequences of instructions which, when executed by a processor, cause the processor to perform a method, the method comprising:

(i) an input print data stream (2) is read in,

(ii) this is analyzed by means of a parser (4) for graphically representable objects (5) and is split up into these graphically representable objects (5), and

(iii) the graphically representable objects (5) are stored in a memory (6) in an object-oriented format,

(iv) the graphically representable objects (5) stored in the memory (6) in an object-oriented format are transformed into a format for the control of an output device (9), preferably a printer, and

(v) the objects thus transformed are combined into an output print data stream (10) and are output,

characterized in that graphically representable objects (5, 5a) are stored in the memory (6) in an object-oriented format, to which at least one stored script is assigned, which is executed in the cases defined in the script.

39. The computer data signal as claimed in claim 38, the method characterized in that the graphically representable objects (5, 5a) are combined into super-objects of higher complexity before being stored in the memory (6).

40. The computer data signal as claimed in claim 38, the method characterized in that feedback messages (11) referring to the output print data stream (10) output are read in and are analyzed for error messages which indicate that the output device (9), preferably the printer, has recognized a transformed graphic object in the output print data stream (10) which cannot be output by said printer,

this graphic object is then split up into part objects of lower complexity, and

the part objects thus obtained, in the format for the control of the output device (9), are slipped into the output print data stream (10) which is output to the output device (9).

41. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which controls external devices, preferably archiving devices, folding systems, enveloping systems or security equipment.

42. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically receives data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

43. The computer data signal as claimed in claim 42, the method characterized in that the script (5a) automatically receiving data also requests this data automatically.

44. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which automatically sends data, preferably data organized in an object-oriented manner, image data, text data or data from web pages from the Internet, data from XML documents or else e-mails.

45. The computer data signal as claimed in claim 44, the method characterized in that the script (5a) sends the graphic object (5) associated with itself to a receiver.

46. The computer data signal as claimed in claim 45, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated

with it, and forwards the graphic object (5) associated with itself to a receiver together with the data requested, received and reassigned by itself.

47. The computer data signal as claimed in claim 42, the method characterized in that the script (5a) in turn reassigns the data received by it to the graphic object (5) associated with itself, and prints out the graphic object (5) assigned to itself together with the data requested, received and reassigned by itself.

48. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a) which is executed in the case of the output of the object (5) defined in the script (5a).

49. The computer data signal as claimed in claim 38, the method characterized in that at least one graphically representable object (5) stored in the memory (6) in the object-oriented format is assigned at least one script (5a), at least one case relating to the execution of the script (5a) being defined in the respective script (5a), and occurring automatically, preferably without further influence from outside.

50. The computer data signal as claimed in claim 49, the method characterized in that the automatically occurring case, defined at least in the respective script (5a), relating to the execution of the script (5a) is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time.

51. The computer data signal as claimed in claim 50, the method characterized in that the timer operates cyclically, that is to say it starts itself again upon expiry.

52. The computer data signal as claimed in claim 38, the method characterized in that Java Script is used as a formal language for the scripts.

53. The computer data signal as claimed in claim 38, the method characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

PaloAlto/59742.1

53. The computer data signal as claimed in claim 38, the method characterized in that the graphically representable objects (5) stored in the memory (6) in an object-oriented format, preferably also script objects (5a), preferably before they are output in the output print data stream (10), are kept ready by an application interface (7) to be read out, to be changed, to be deleted or to have new objects (5) appended.

PaloAlto/60034.1

UNITED STATES PATENT AND TRADEMARK OFFICE

SEPTEMBER 08, 2003

PTAS

SQUIRE, SANDERS & DEMPSEY, L.L.P.
AARON WININGER, ESQ.
600 HANSEN WAY
PALO ALTO, CA 94304-1043

Deputy Under Secretary of Commerce For Intellectual Property and
Deputy Director of the United States Patent and Trademark Office
Washington, DC 20231
www.uspto.gov

102419756A

UNITED STATES PATENT AND TRADEMARK OFFICE
NOTICE OF RECORDATION OF ASSIGNMENT DOCUMENT

THE ENCLOSED DOCUMENT HAS BEEN RECORDED BY THE ASSIGNMENT DIVISION OF THE U.S. PATENT AND TRADEMARK OFFICE. A COMPLETE MICROFILM COPY IS AVAILABLE AT THE ASSIGNMENT SEARCH ROOM ON THE REEL AND FRAME NUMBER REFERENCED BELOW.

PLEASE REVIEW ALL INFORMATION CONTAINED ON THIS NOTICE. THE INFORMATION CONTAINED ON THIS RECORDATION NOTICE REFLECTS THE DATA PRESENT IN THE PATENT AND TRADEMARK ASSIGNMENT SYSTEM. IF YOU SHOULD FIND ANY ERRORS OR HAVE QUESTIONS CONCERNING THIS NOTICE, YOU MAY CONTACT THE EMPLOYEE WHOSE NAME APPEARS ON THIS NOTICE AT 703-308-9723. PLEASE SEND REQUEST FOR CORRECTION TO: U.S. PATENT AND TRADEMARK OFFICE, ASSIGNMENT DIVISION, BOX ASSIGNMENTS, CG-4, 1213 JEFFERSON DAVIS HWY, SUITE 320, WASHINGTON, D.C. 20231.

RECORDATION DATE: 11/07/2002

REEL/FRAME: 013942/0250
NUMBER OF PAGES: 3

BRIEF: ASSIGNMENT OF ASSIGNOR'S INTEREST (SEE DOCUMENT FOR DETAILS).

ASSIGNOR:
KRAUTTER, THOMAS

DOC DATE: 10/05/2002

ASSIGNEE:
CCP SYSTEMS AG
HELLMUTH-HIRTH-STR. 9
STUTTGART, FED REP GERMANY 70435

SERIAL NUMBER: 10275784
PATENT NUMBER:

FILING DATE: 11/07/2002
ISSUE DATE:

DATES ENTERED

NO Action

THERESA FREDERICK, EXAMINER
ASSIGNMENT DIVISION
OFFICE OF PUBLIC RECORDS

SEP 1 1 2003

CALENDARED

10/275784

A/9 581 - uc

DT12 Rec'd PCT/PTO 07 NOV 2002

EXPRESS MAIL LABEL NO.: EL 701 318 593 US Attorney Docket No. 60428.00001

Form PTO-1595 (Rev. 10/02) OMB No. 0651-0027 (exp. 6/30/2005)

RECOR[illegible]

04-15-2003

U.S. DEPARTMENT OF COMMERCE
U.S. Patent and Trademark Office

Tab settings ⇨ ⇨ ⇨ ▼ ▼ ▼ ▼ ▼

102419756

To the Honorable Commissioner of Pa[illegible] ...l original documents or copy thereof.

1. Name of conveying party(ies):
Thomas Krautter

11-7-02

Additional name of conveying party(ies) attached? ☐ Yes ☒ No

2. Name and address of receiving party(ies)

Name: CCP SYSTEMS AG

Internal Address:

Street Address: Hellmuth-Hirth-Str. 9

City: Stuttgart Country: Germany Zip: 70435

Additional Name(s) & address(es) attached? ☐ Yes ☒ No

3. Nature of conveyance:

☒ Assignment ☐ Merger

☐ Security Agreement ☐ Change of Name

☐ Other _____

Execution Date: October 5, 2002

4. Application number(s) or patent number(s):

If this document is being filed together with a new application, the execution date of the application is: October 5, 2002

A. Patent Application No.(s)

1027S784

B. Patent No.(s)

Additional numbers attached? ☐ Yes ☒ No

5. Name and address of party to whom correspondence concerning this document should be mailed:

Name: Aaron Wininger, Esq.

Internal Address: Squire, Sanders & Dempsey, L.L.P.

11/14/2002 SNAJARRO 00000132 050150 10275784

04 FC:8021 40.00 CH

Street Address: 600 Hansen Way

City: Palo Alto State: CA Zip: 94304-1043

6. Total number of applications and patents involved: 1

7. Total fee (37 CFR 3.41) . . . .$ 40.00

☐ Enclosed

☒ Authorized to be charged to deposit account

8. Deposit account number:

05-0150

(Attach duplicate copy of this page if paying by deposit account)

DO NOT USE THIS SPACE

9. Statement and signature.

To the best of my knowledge and belief, the foregoing information is true and correct and any attached copy is a true copy of the original document.

Aaron Wininger, Reg. No. 45,229 [signature] November 7, 2002

ASSIGNMENT

(1-4) Insert Name(s) of Inventor(s)	(1) Thomas Krautter	(3) ____________
	(2) ____________	(4) ____________

For good and valuable consideration receipt of which is hereby acknowledged, the undersigned agree(s) to assign, and hereb do(es) assign, transfer and set over to:

(9)	*Insert name of Assignee*	(9) CCP SYSTEMS AG
(10)	*Insert state of incorporation of Assignee*	(10) Germany
(11)	*Insert address of Assignee*	(11) of Hellmuth-Hirth-Str. 9, 70435 Stuttgart, Germany (hereinafter designated as the Assignee) the entire worldwide right, title, interest, a patent applications and patents for every country, including divisions, reissues, continuations and all other extensions, rights and priorities in the invention known and subject matter contained in
(12)	*Insert Identification of Invention, such as Title, Case Number or Foreign Application Number*	(12) Method And System For The Transformation Of Digital Print Data Strean And Corresponding Printer And Printer Server for which the undersigned has (have) executed an application for patent in United States of America
(13)	*Insert Date of Signing of Application*	(13) on October 5, 2002

1) The undersigned agree(s) to execute all papers necessary in connection with the application and any continuing division applications thereof and also to execute separate assignments in connection with such applications as the Assignee r deem necessary or expedient.

2) The undersigned agree(s) to execute all papers necessary in connection with any interference which may be declared concerning this application or continuation or division thereof and to cooperate with the Assignee in every way possible in obtaining evidence and going forward with such interference.

3) The undersigned agree(s) to execute all papers and documents and perform any act which may be necessary in connection with claims or provisions of the International Convention for Protection of Industrial Property or similar agreeme

4) The undersigned agree(s) to perform all affirmative acts which may be necessary to obtain a grant of a valid United States patent to the Assignee.

5) The undersigned hereby authorize(s) and request(s) the Commissioner for Patents and the duly constituted authorities of foreign countries to issue any and all Letters Patents resulting from said application or any division or division continuing or reissue applications thereof to the said Assignee, its successors and assigns, as Assignee of the entire right, title and interest, and hereby covenants that he has (they have) full right to convey the entire interest herein assigned, and that he (they have) not executed and will not execute, any agreement in conflict herewith.

6) The undersigned hereby grant(s)

Marc A. Sockol, Reg. No. 40,823; Vidya R. Bhakar, Reg. No. 42,323; Daryl C. Josephson, Reg. No. 37,365; Cameron Kerrigan, Reg. No. 44,826; David B. Abel, Reg. No. 32,394; Nathan Lane, Reg. No. 43,738; Michael Lechter, Reg. No. 27,350; David Koo, Reg. No. 46,839; David Rogers, Reg. No. 38,287; William Bachand, Reg. No. 34,980; Aaron Wininger, Reg. No. 45,229; Paul A. Durdik, Reg. No. 37,819; Paul J. Meyer 47,791; Reg. No. 48,821; Victor Repkin, Reg. No. 45,039; Victoria L. Nicholson, Reg. No. 47,823; and Fariba Sirjani, Reg. No. 47,947.

the power to insert on this assignment any further identification which may be necessary or desirable in order to comply wit the rules of the United States Patent and Trademark Office for recordation of this document.

Date: 5.10.2002

Thomas Krautter

EXHIBIT C

SYSTEM p SOFTWARE
23, 2009
ed by Michelly Paula Mantovani
ON YOUR ADOBE VERSION, LOOK FOR THE BINOCULAR ICON OF FOR THE WORD "FIND."
PRODUCT NUMBER AND THE SCREEN WILL SCROLL TO THE APPROPRIATE SECTION IN THE DOCUMENT.
ntract, IBM's software is expected to be sold with hardware.
the addition of staff or a roll out would allow for separate purchases.
licy of linking hardware with software sale can be directed to the WSCA/NASPO
Bernie Kopischke at Bernie.Kopischke@state.mn.us

subject to change and do not contain any State & Local taxes,
Please consult your IBM representative for current prices or
exemptions

ict families may vary and some products eligible for sale
oduct guidelines may have a zero discount

on.
e without notice.
e for exact configuration/pricng.

ITELLIGENT SOFTWARE V1.2

CRIPTION	LIST PRICE	WSCA Discount	WSCA Price
LLIGENT SOFTWARE	$0.00	0.15	$0.00
USERS W/1Y SWM	$8,000.00	0.15	$6,800.00
000 W/1 YR SWMA	$12,000.00	0.15	$10,200.00
-2499 W/1 Y SWMA	$18,000.00	0.15	$15,300.00
+ W/ 1 Y SWMA	$27,000.00	0.15	$22,950.00

5 NETWORK E-MAIL SECURITY EXPRESS S(

CRIPTION	LIST PRICE	WSCA Discount	WSCA Price
EMAIL EXP SOL SYS P	$0.00	0.15	$0.00
/ 1Y SUB+SPT	$5.00	0.15	$4.25
W 1YR SUB+SPT	$4.00	0.15	$3.40
R MPP 1Y SUB+SPT	$3.00	0.15	$2.55
/ 1Y SUB+SPT	$2.00	0.15	$1.70

Appendix C

IBM

Dictionary of Computing

▼ The most comprehensive computing dictionary ever published

▼ More than 18,000 entries

IBM DICTIONARY OF COMPUTING

Compiled and edited by

GEORGE McDANIEL

McGRAW-HILL, INC.
New York San Francisco Washington, D.C. Auckland Bogotá
Caracas Lisbon London Madrid Mexico City Milan
Montreal New Delhi San Juan Singapore
Sydney Tokyo Toronto

Limitation of Liability

While the Editor and Publisher of this book have made reasonable efforts to ensure the accuracy and timeliness of the information contained herein, neither the Editor nor the Publisher shall have any liability with respect to loss or damage caused or alleged to be caused by reliance on any information contained herein.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 9 9 8 7 6 5 4 3

ISBN 0-07-031488-8 (HC)
ISBN 0-07-031489-6 (PBK)

The sponsoring editor for this book was Daniel A. Gonneau and the production supervisor was Thomas G. Kowalczyk.

Printed and bound by R. R. Donnelley & Sons Company.

Tenth Edition (August 1993)

This is a major revision of the *IBM Dictionary of Computing,* SC20-1699-8, which is made obsolete by this edition. Changes are made periodically to the information provided herein.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. Comments may be addressed to IBM Corporation, Department E37/656, P. O. Box 12195, Research Triangle Park, NC 27709.

International Edition

When ordering this title, use ISBN 0-07-113383-6.

This book is printed on acid-free paper.

er numbers when the
as numeric only. The
ric keys on an adding

d in one of the punch
ı nine. A zero-punch,
ıe-punch, in combina-
ı, is considered a zone

crete representation of

selecting the numeric
keyboard-printer.

P) A word processing
proportionally spaced
the active position to
qual to the escapement
being used. See also

of a numeric data item
ıanged during program

nsists of digits and pos-
pecial characters; for
imal Classification, the
sed as an identifier for
ıglish. (T)

ɔard that a user presses
meric keypad so that it
y pressed.

access memory.

.

station.

ibble. Deprecated term
ıalfbyte.

ıe highest frequency of
ɔrrectly sampled. The
lf of the sampling fre-

O

OA Office automation. See automated office.

OACBRU Open ACB request/response unit.

OAF Origin address field.

OAF' Origin address field prime.

OAM Storage Management Component In the Object Access Method, the component that determines where objects should be stored, manages object movement within the object storage hierarchy, and manages expiration attributes based on the installation storage management policy.

OAR Operator authorization record.

object (1) In computer security, anything to which access is controlled; for example, a file, a program, an area of main storage. (2) A passive entity that contains or receives data; for example, bytes, blocks, clocks, fields, files, directories, displays, keyboards, network nodes, pages, printers, processors, programs, records, segments, words. Access to an object implies access to the information it contains. (3) In SAA Common User Access architecture, something that a user works with to perform a task. Text and graphics are examples of objects. (4) In AIX Enhanced X-Windows, a software abstraction consisting of private data and private and public routines that operate on the private data. Users can interact with an object only through calls to the public routines of the object. (5) In the AIX object data manager, an instance or member of an object class, conceptually similar to a structure that is a member or array of structures. See also object class. (6) In programming languages, a data object. (7) In AIX graphics, synonym for display list. (8) In the Network Computing System, an entity that is manipulated by well-defined operation; for example, a disk, a file, a printer. Every object has a type and is accessed through an interface. (9) In the IBM ImagePlus system, a collection of structured fields. The first structured field provides a begin-object function and the last structured field provides an end-object function. The object may contain one or more other structured fields whose content consists of one or more data elements of a particular data type. An object may be assigned a name, which may be used to reference the object. Examples of objects are text, font, graphics, image, and formatted data objects. (10) In object-oriented design or programming, an abstraction consisting of data and the operations associated with that data. See also class. (11) In the AS/400 system, a named storage space that consists of a set of characteristics that describe itself and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed; for example, programs, files, libraries, and folders. (12) In SQL, anything that can be created or manipulated with SQL statements, such as databases, tables, views, or indexes. (13) See arithmetic object, compound object, data object, integral object.

Object Access Method (OAM) In the IBM ImagePlus system, a program that provides object storage, object retrieval, and object storage hierarchy management. The Object Access Method isolates applications from storage devices, storage management, and storage device hierarchy management.

object-action In SAA Common User Access architecture, a process sequence in which a user selects an object and then selects an action to apply to that object. Contrast with action-object.

object authority (1) In the AS/400 system, a specific authority that controls what a system user can do with an entire object; for example, object authority includes deleting, moving, or renaming an object. There are three types of object authorities: object existence, object management, and object operational. (2) The right to use or control an object. See also data rights, object rights.

object class A categorization or grouping of objects that share similar behaviors and circumstances.

object code Output from a compiler or assembler which is itself executable machine code or is suitable for processing to produce executable machine code. (A)

object code compatibility (1) Pertaining to object programs that can be run on two or more systems without recompilation or reassembly. (2) Contrast with source code compatibility.

OBJECT-COMPUTER In COBOL, the name of an Environment Division paragraph in which the computer environment, within which the object program is executed, is described.

object-computer entry In COBOL, an entry in the OBJECT-COMPUTER paragraph of the Environment Division that contains clauses that describe the computer environment in which the object program is to be executed.

Appendix D

APPENDIX D

'789 patent Classifications	Interleaf Patent Classifications
101/484 **PRINTING PROCESSES**	717/139: **DATA PROCESSING**: Software Development, Installation, And Management Software Program Development Tool (E.G., Integrated Case Tool Or Stand-Alone Development Tool)
400/61, 62, 63; 400/76: **TYPEWRITING MACHINES i**ncluding Control Of Format And Selection Of Type-Face By Programmed Control-System Including Control Of Format By Programmed-Control-System	715/234: **DATA PROCESSING**: Presentation Processing Of Document, Operator Interface Processing, And Screen Saver Display Processing Presentation Processing Of Document
358/1.1, 1.9, 1.15, 1.16, 1.17, 1.18 **FACSIMILE AND STATIC PRESENTATION:** Processing Static Presentation Processing (E.G., Processing Data For Printer, Etc.)	

Attorney Docket No.: P-74637-US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patentee(s):	KRAUTTER, Thomas	Examiner:	STEELMAN, Mary J.
Control No.:	95/001,398	Reexamination Filed:	July 16, 2010
Patent No.	6,684,789	Art Unit:	3992
Title:	METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER		

DECLARATION OF CHRISTOPH PICHT

I, Christoph Picht, declare as follows:

1. I am Director of Business Development at CCP Systems AG ("CCP"). CCP owns U.S. Patent No. 6,684,789 ("the '789 patent") that is the subject of the current reexamination. I started working at CCP in April 2006.

2. I received a computer science degree (Diplom Informatiker) from Technical High School in Darmstadt, Germany, in 1981.

3. I have been working in software development since then.

4. I have been working in the electronic printing business for about 24 years.

5. I make this declaration in support of CCP's Response to the November 19, 2010 Office Action in the reexamination identified above.

6. I have read and am familiar with the '789 patent.

7. CCP and IBM entered into a license in July 2004 that gave IBM a "license under any patents … to make, have made, use [etc.] ..." that included the '789 patent.

8. CCP has produced software called JScribe®. I am familiar with the functionality of CCP's JScribe® product.

9. JScibe® is a software product that CCP had licensed to IBM Corporation's German subsidiary. IBM's Korean subsidiary in turn sublicensed the JScribe® software to Samsung Electronics Corp., Ltd., Korea.

10. Samsung has incorporated the JScribe® software or parts of it into many of its printers and multifunctional devices.

11. In my view, Claim 1 of the '789 patent covers the JScribe® software when used for printing. JScribe® reads in an input print stream that is analyzed by a parser. This stream is split up into graphically representable objects which are then stored in a memory in an object-oriented

format. JScribe® software and any of its functionalities can be activated by scripts which can be developed easily by users, either with CCP's Software Development Kit (SDK), or with any other editor, and then be deployed to the JScribe® enabled device where they become active.

12. There are also several companies that write scripts that can be used with JScribe® software. Samsung, for example, has appointed several solution partners which use JScribe® software for integrating Samsung devices into their solutions.

13. To the extent that Samsung's printers do not include scripts assigned to objects, Samsung's printers with the JScribe® software installed have an interface that allows users to easily create scripts and to include them in the software.

14. At least one Samsung US customer (identified in Widuch Dec. Ex. E, p. 5), which according to Samsung, has "[o]ver 1000" printers, where the printers also have scripts assigned to an object, including ones that are used for security equipment (see e.g., claims 4, 25, and 41).

15. Similarly, I believe that CCP's JScribe® software, which I understand has been incorporated into Samsung printers, has the functionality to include each of the claim elements of claims 2-16 of the '789 patent. That is, users can create scripts that interface with the JScribe® software so that they include the ability to create, or function, as follows:

a. super-objects, as claimed in Claim 2;

b. feedback messages, as claimed in Claim 3;

c. control over external devices, as claimed in Claim 4;

d. automatically to receive data, as claimed in Claim 5;

e. automatically to make data requests, as claimed in Claim 6;

f. reassign data and print out the graphic objects, as claimed in Claim 7;

g. automatically send data, as claimed in claim 8;

h. send graphic objects, as claimed in Claim 9;

i. reassign data to graphic objects and print out the graphic object, as claimed in Claim 10;

j. at least one graphically representable object is assigned to at least one script, which is executed in the case of the output of the object defined in the script, as claimed in Claim 11;

k. at least one graphically representable object is assigned at least one script, at least one case relating to the execution of the script being defined in the respective script, and occurring automatically, as claimed in Claim 12;

l. in an automatically occurring case, defined in the respective script, relating to the execution of the script, a timer, so that a case occurs automatically as a result of expiry of time, as claimed in Claim 13;

m. the timer operates cyclically, that is, it starts itself again upon expiry, as claimed in Claim 14;

n. Java Script is used as a formal language for the scripts, as claimed in Claim 15; and

o. objects stored in memory can be changed or deleted or have new objects appended, as claimed in Claim 16.

16. In addition, as explained above, systems, like various Samsung printers which include the JScribe® software, that are connected to a computer network are covered by claims 17-19.

17. For similar reasons, I believe that Samsung's printers which include JScribe® software, are covered by claim 20.

18. As explained in paragraph 15 above, at least one Samsung US customer has a print server that includes JScribe® software. There are at least half a dozen other companies that have JScribe® software loaded on their print servers. All have scripts assigned to objects that perform various functions, including, for example, having timers and the ability to reassign data. (See e.g., claims 7, 13 and 21).

19. Samsung printers, which include the JScribe® software, also are covered by claims 22-37 because they have a memory (i.e., a computer readable medium) on which the infringing method steps are stored. Scripts that users install on the printer would also be stored in memory, and these method steps would be carried out on the printer processor when the execution command is triggered (e.g., circumstances defined in the script, for example, a timer).

20. I have reviewed the proposed new claims that CCP has submitted in this reexamination, and believe the new claims CCP has presented here cover CCP's JScribe® technology as discussed above.

21. In addition to the '789 patent, CCP obtained corresponding patents in Europe (EP 1282883 B1) and Japan (JP 3974782 B2).

I declare under penalty of perjury under the laws of the United States of America that all statements made here are based on my own knowledge and are believed to be true. I understand that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. section 1001), and may jeopardize the validity of the patent subject to reexamination.

29.03.2011
Date

Christoph Picht
Director of Business Development
CCP Systems AG

Attorney Docket No.: P-74637-US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patentee(s):	KRAUTTER, Thomas	Examiner:	STEELMAN, Mary J.
Control No.:	95/001,398	Reexamination Filed:	July 16, 2010
Patent No.	6,684,789	Art Unit:	3992
Title:	METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER		

AMENDED DECLARATION OF DAVID BIRNBAUM, PH.D.

I, David Birnbaum, declare as follows:

1. I hold M.A. and Ph.D degrees in Physics from the University of Rochester. I received a B.Engr. in Engineering Physics from Cornell University. My resume is attached as Exhibit A.

2. From 1980 until 2003, I worked at Xerox Corporation, most recently as an Engineering Fellow and Principal Scientist where I was responsible for the design and development of printers and printing systems.

3. I have fifteen (15) U.S. patents granted to date, as well as foreign counterpart patents, and have published numerous articles in scientific journals. I have also passed the examination for admission to practice in the United States Patent Office as a Patent Agent.

4. My awards and professional affiliations are summarized in my resume.

5. I am familiar with the subject matter of U.S. Patent No. 6,684,789 (the "'789 patent"), including printing software and print stream processing.

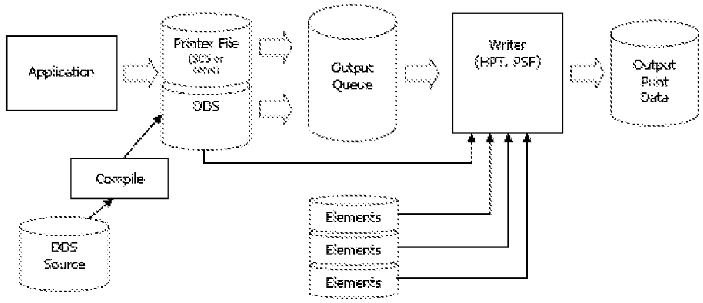
6. I have reviewed the '789 patent, the Office action, and the cited references.

7. I am making this declaration, having been hired by CCP Corporation in the matter of the reexamination of the '789 patent. I am being compensated at my normal hourly rate.

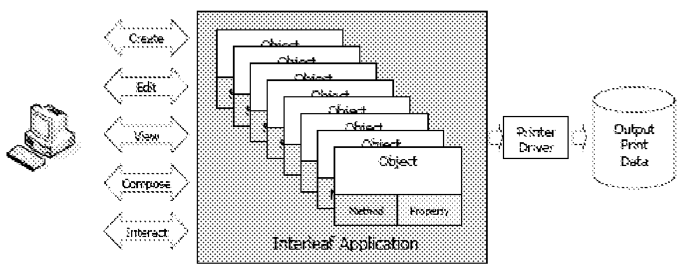
Brief Summary of the Cited References

8. IBM's AS/400 is a midrange computer system that supports multiple users. IBM AS/400 Guide to Advanced Function Presentation and Print Services Facility ("IBM") describes Advanced Function Presentation ("AFP"). AFP allows for graphical presentation of data produced by applications running on the AS/400. The AS/400 receives data from an application (including text, images, etc.), prepares the data (including, for example, by executing a DDS (Data Description

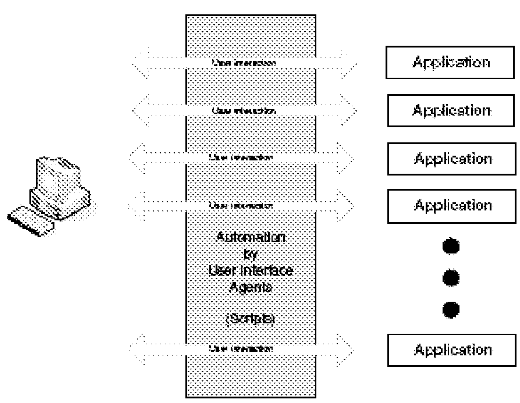
Specification) on the input print data) as output print data, and sends this output data to the selected printer. The following schematic figure describes the data flow in the AS/400 system:



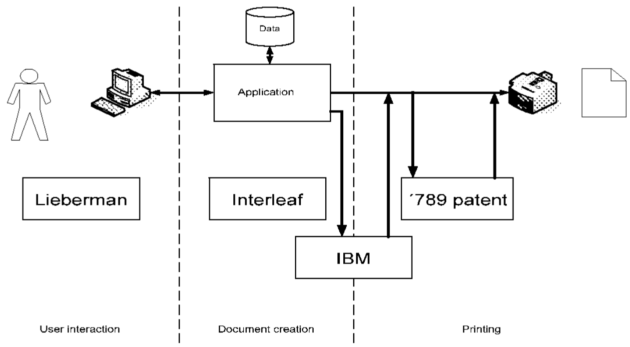
9. Interleaf Active Documents ("Interleaf") and U.S. Patent No. 5,579,519 ("Interleaf Patent") describe creation of a so-called "active documents" (collectively, the "Interleaf documents"). "Active documents" are created at the computer, far upstream from the data print stream discussed in IBM (or the '789 patent). Documents in the Interleaf references may certainly be printed; however, the only print stream is the output of a print process, and the treatment of this print stream is not discussed in detail in the Interleaf documents. The following schematic figure describes the Interleaf documents:



10. The article "Integrating User Interface Agents with Conventional Applications", by Henry Lieberman ("Lieberman"), describes automating user interaction by using scripts. The following schematic figure describes Lieberman:



11. The following schematic diagram shows the location in the document creation and printing "food chain" where these various references may be found:



I. Ground of Rejection No. 1: IBM

A. Parser

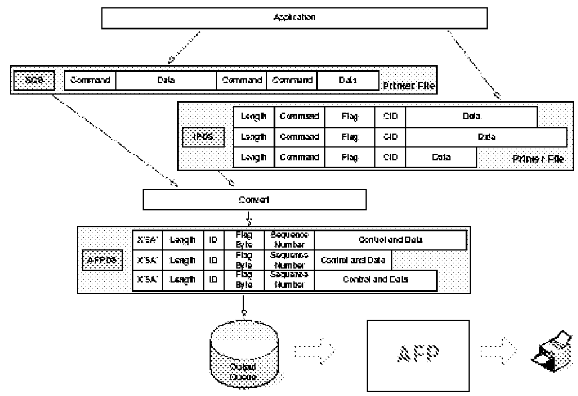
12. A parser, as properly understood according to the '789 patent, is a syntax analyzer: it is capable of analyzing the syntax of an input data stream based on a formal grammar.

13. The "parser" in the '789 patent is consistent with a well-accepted meaning of the term in computer science. This functionality permits the parser to analyze even complex page description languages (PDLs), in comparison to a state machine driven by regular expressions, which can only reasonably handle single commands in a line by line approach.

14. The function described in IBM at pp. 194-195 is capable of being performed by a state machine, and would not require a parser as claimed in the '789 patent.

15. By referring to the data fields described at IBM, pp. 14-15, it is clear that input print lines are composed of structured fields. Identifying and separating the individual fields from a structured field is simpler than parsing, and can be performed by a state machine.

16. The input process described in IBM involves receiving lines in SCS (SNA [System Network Architecture] Character Stream), or IPDS (Intelligent Print Data Stream) format, and converting data fields in such lines into respective fields in AFPDS (Advanced Function Print Data Stream). Both SCS and IPDS have structured formats. The following figure describes the input process in IBM, described in further detail below:



17. The SCS data stream consists of a one-byte hexadecimal code followed by the data to be printed. SCS is used to control line printers and supports row and column functions. The IBM iSeries Printer Device Programming, Version 5, Document No. SC41-5713-05, published in 2002 (cover page, copyright page, table of contents iii-viii and 492-98 attached as Exhibit B and Ex. C Version 4 at 556), describes IPDS as follows: "[t]he structured field format of IPDS allows commands to be sent to the printer in a continuous stream. Each command is self-describing." (Ex. B at 492, and Ex. C at 555-56, emphasis added). That is, IPDS has no grammar and no syntax across commands.

The IPDS Command Format

All IPDS commands are encoded in the following format:

Length	Command	Flag	CID	Data

18. It is in this context of the structured format of SCS and IPDS that the cited statement in IBM must be understood. The "input print lines" are made up of structured data fields, as explained at IBM, pp. 14-15. It should be noted that page description languages (PDL) do not have "input print lines" and therefore, the statement in the IBM reference about "parsing" is not applicable to PDL input data streams, including Postscript. There is absolutely no disclosure in IBM on whether (and, if so, how) PDL inputs are "parsed" within any reasonable understanding of the term as used in the patent claims.

19. The result of the conversion is data in AFP data structure (AFPDS) format, which is another structured field format, as shown in IBM Fig. 3, at p. 21.

20. A field in AFPDS is identified by a leading 0X5A byte, followed by fields identifying the length of the data, the type of data "ID", a flag byte and a sequence number, followed by the data. The location and content of the structured fields are predefined and do not require parsing. The meaning of the values in each field also is predefined. Thus, evaluation and processing of an input

print line in IBM, which is defined as a structured field, can be done by a series of if-then statements, which is a typical implementation of a state machine.

21. Such an evaluation is one implementation of a state machine, which the ‘789 patent expressly distinguishes from the parser used in the invention. (See Col. 3, lines 21-30).

22. In contrast to IBM, as explained above, parsing in the ‘789 patent is complex: it requires scanning the input data stream character by character, identifying tokens, e.g., collections of characters that correspond to commands or data and a syntax evaluation to determine the relationships between the identified tokens. Often there is no rigid format for the input data stream. Even though IBM AS/400 can handle Postscript documents as an input – it cannot parse them into graphically representable objects. This is because the only methodology disclosed in IBM requires conversion into structured fields. (See e.g., IBM p. 194: input print lines are “parsed into individual fields”).

23. IBM therefore does not disclose a parser as defined and claimed in the ‘789 patent. The cited portion of IBM only discloses conversion from structured SCS or IPDS format to AFPDS. That function is not parsing under any reasonable interpretation of the ‘789 patent claims, because it does not require syntax analysis.

B. Object-Oriented Format

24. In computer science, “object-oriented programming” uses objects – i.e., data, properties and methods, together with their interactions as defined by the methods.

25. Object-oriented programming techniques typically include features such as class hierarchy, data abstraction, encapsulation, modularity, polymorphism, and inheritance.

26. An object-oriented format in the sense of the ‘789 patent, and as generally understood in computer programming, requires that objects have the ability to be organized, for example, in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend.

27. IBM uses the term “object” to refer to data structures in which various document elements (e.g., image, form, text, font, bar code, and graphic) are represented. For a number of reasons -- including lack of: class hierarchy, assignment of methods to objects and inheritance -- “object,” as used in IBM, does not have the same meaning as “object” in object-oriented programming, and as that term is used in the ‘789 patent.

28. For example, page 23 of IBM purports to describe as an “object” an image of an airplane. However, that “object” is not handled as a member of a class to which dedicated methods are assigned – and thus cannot be considered an object in the object-oriented programming sense.

29. IBM discloses simple data structures whose data are treated as a unit – and nothing more. (See, e.g., Ex. B at p. 495; Ex. C at 558). The structured field format of AFPDS is suitable for storing such data elements, but not objects in object-oriented format.

30. By contrast, an object-oriented format, in the sense of the '789 patent and elsewhere, requires that objects be capable of being organized in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend.

C. Scripts

31. A script is a program or series of commands that is interpreted and runs in real time rather than compiled and then executed.

32. Scripts are defined as: "[a] high-level programming, or command, language that is interpreted (translated on the fly) rather than compiled ahead of time." And, "[s]cripts are interpreted as they are run." See Exhibit D.

33. In contrast to scripts, programs written in standard program languages, such as FORTRAN, C++, etc., must typically be compiled into object code before being executed.

34. Using scripts provides at least two advantages: flexibility and portability.

35. Scripts may be freely modified, including at run-time, which is not true for compiled code. For example, in one embodiment of the invention, the data processing unit permits stored objects, including particularly script objects, to be read out graphically, to be changed, to be deleted or to be appended. See, e.g., claims 18, 19. This cannot be done using compiled object code.

36. Object code is limited because it is platform-dependent. A program written in a compiled language must be compiled into object code for a particular processor architecture. In contrast, any processor capable of interpreting a script can run the script. This renders scripts portable, i.e., platform-independent.

37. This means the DDS in IBM is compiled before being used for printing, can only be executed as object code after being compiled, and, unlike scripts, must be recompiled if modified.

38. This feature of DDS is a critical distinction from a script, in which interpretation and execution of the script are typically simultaneously triggered, in the case of the '789 patent, for example, by the incoming print data.

D. Script Assigned to an Object

39. Even assuming that the "elements" in IBM (images, text, etc.) were objects stored in object-oriented format, and that the DDS used in IBM is a script, the DDS is not assigned to an element, but rather is applied to the document as a whole.

40. As described in the '789 patent, by assigning a script to a particular object, various benefits and advantages may be obtained. (See e.g., col. 5, lines 27-31 and col. 6, lines 38-51).

41. The architecture of the AS/400 described in IBM would have to be entirely reconfigured and the code totally rewritten to perform these operations.

E. The Claimed System in a Printer

42. In addition to the above, claim 20 recites a "printer, characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17."

43. The AS/400 is not a printer, and a printer connected to the AS/400 would not have in it a system for the transformation of digital print data streams, nor would such a printer have the capability to compile or execute DDS.

44. It would not have been obvious to take the functionality of the AS/400 described above (a midrange computer) and import it into a printer, insofar as doing so would be contrary to the centralized organization of the AS/400 system, and a technological impossibility.

F. Dependent Claims

45. Regarding claims 2, 23 and 39, nothing in IBM suggests that the data structures described therein (e.g., an image of an airplane) are linked to other objects with the properties listed above (class hierarchy, assignment of methods to objects, and inheritance).

46. Because the IBM elements do not possess these properties, they cannot be linked together to form super-objects with those properties.

47. Documents and pages as used in IBM are not objects in the object-oriented sense, as claimed in the '789 patent. Thus, they cannot be combined into a super-object.

48. Similarly, lines, arcs, etc., are not objects, for reasons explained above, and combining them into a geometric shape does not create a super-object within the meaning of the '789 patent claims.

49. Regarding claims 3, 24, and 40, the feedback mechanism provided by IPDS does not have the functionality recited in these claims.

50. IBM does not disclose analysis of feedback messages for error messages that indicate that the output device has recognized a transformed graphic object in the output print data stream which cannot be output by the printer, and based on such a feedback message, splitting up the graphic object into objects of lower complexity that can be printed.

51. Regarding claims 4, 25, and 41, as discussed above, IBM does not disclose scripts, scripts assigned to objects, or objects stored in object-oriented format. Thus, IBM as a whole cannot

disclose that at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which controls external devices, as recited in those claims.

52. Regarding claims 5 and 26, none of the cited portions of IBM discuss automatically receiving data organized in an object-oriented manner, e.g, from the Internet, or e-mails.

II. Ground of Rejection No. 2: IBM in view of Interleaf

A. The Field of the Invention and the Level of Skill

53. The field of the invention of the '789 patent is printing software generally, and in particular, interpretation or processing of print data streams at a printer or print server.

54. To the best of my knowledge, prior to 2000, there were no discrete fields as Samsung has described them, namely: (1) "object oriented document publishing systems with scripting functionality;" or (2) "scripting interfaces for object systems."

55. A person of ordinary skill working in the relevant field (i.e., printing software generally, and in particular, interpretation or processing of print data streams at a printer or print server) in May 2000 or May 2001, would have been a person with a computer science degree (typically, a bachelor's degree), and approximately 2-4 years of experience.

56. This experience may typically have been acquired working at a company making printers or printer software, such as Xerox, Hewlett-Packard, etc.

57. People involved in document creation software and those working in printer software or programming would typically have been working in different divisions or departments, or more likely in different companies altogether.

58. IBM describes how data from an application running on the AS/400 may be formatted (e.g., using a DDS) and sent to a printer. It describes providing print capability to networks controlled by a central server, like the AS/400.

59. The Interleaf documents deal with "document creation" and manipulation software (Interleaf, p. 75). These fields are unrelated to the field of the invention of the '789 patent.

60. The print function described (briefly) in the Interleaf documents is irrelevant to what happens to the print data downstream. That is, the Interleaf documents do not involve printer software or interpretation of the print data streams at a printer or print server.

61. Understood in the proper context, the Interleaf documents are in an entirely different field of technology than the '789 patent. One of ordinary skill in the art would not have referred to the Interleaf documents or combined them with IBM to arrive at the inventions of the '789 patent.

B. No Motivation to Combine the IBM and Interleaf Documents

62. The Office action stated that "[c]ombining the print process of IBM that parses and transforms a print stream into an object-oriented format, and the print-scripting process of Interleaf are combining well known printing techniques to yield predictable results."

63. The phrase "print-scripting process of Interleaf" is unclear and not disclosed in the Interleaf documents.

64. Conceivably, after extensive redesign, the Interleaf and IBM references might have been linked, but not combined. For example, a user *might* be able to operate the Interleaf system on the PC, and then send a document to the AS/400 to print. The AS/400 would then process the data, for example, by formatting it using a DDS. The result, however, would not result in the inventions claimed in the '789 patent because any object-oriented formatting, including any scripts, in the Interleaf "active document" would be lost prior to generating the print data stream that would be transmitted to the AS/400. Thus, even in a "linked" system, the AS/400 print data stream could not have either: (a) object-oriented formatting or (b) scripts disclosed in the Interleaf references.

65. A cited motivation for combining IBM and Interleaf is that "Interleaf suggests the combination because it is explicitly designed to leverage outside printing and publishing functionality," citing to Interleaf, p. 82, which refers to external publishing functionality, such as printing (Office action, p. 11). In a "combined" IBM+Interleaf system, Interleaf documents may be formatted and then sent to the AS/400. However, in order to combine the features of Interleaf and IBM, one would have had to redesign the AS/400 AFP by taking the entire object-oriented structure, together with the scripts, described in Interleaf, and import them onto the AS/400, where the printing facilities reside. There is no suggestion or reason to do this. It would result in an entirely reconstructed system.

66. Another purported motivation is that "IBM provides external printing and publishing functionality. . . IBM also teaches the desire to control outside printing and publishing functionality. . ." (Office action, pp. 11-12). The "external" functionality of Interleaf refers to printing, whereas the "external" functionality of IBM refers to staplers and binders, neither of which requires scripting functionality or object oriented code.

67. In addition, as discussed above, the AS/400 operates on data using a compiled DDS, not a script. Therefore, it would not have been obvious – even in light of Interleaf – to modify the AS/400 to use scripts.

68. The AS/400 is a centralized platform, and therefore, would not have benefitted from the platform-independence of scripts.

69. On the contrary, scripts may often be slower to execute than object code, because they must be interpreted in real time. There would have not been any reason to replace the compiled DDS with a script to be interpreted at run time.

C. The IBM and Interleaf References Even if Combined Would Not have Resulted in the Inventions of the '789 Patent

70. Even if the Interleaf system operating on a PC were linked to an AS/400 midrange computer, it would not have resulted in the present invention.

71. A user may operate the Interleaf system on the PC, and then upon entering a "print" command in the application, send the output document to the AS/400 to print. However, the print output of such a PC would immediately have been converted to the structured AFP format in order to be manipulated by the AS/400.

72. As explained above, this conversion into the structured AFP data stream is not the purported "parsing" to which Samsung referred (IBM reference, p. 194). Nor would using the Interleaf software on a PC client connected to an IBM AS/400 cause the AS/400 to store graphically representable objects in object-oriented format, or assign a script to an object. Adding Interleaf to the "front end" of IBM would simply have resulted in printing documents as disclosed in IBM. Nothing in the print stream would have changed: there would still have been no objects, no scripts assigned to objects and no parser.

D. Dependent Claims

73. Regarding claims 2, 23, and 39, Samsung pointed to Interleaf, saying that the reference discloses component objects being changed, for example, "while the document is opening". However, this process of opening a document bears no relation to transforming a print data stream, as claimed in the '789 patent.

74. Regarding claims 4, 25, and 41, Samsung cited two statements in Interleaf: (1) that active documents are defined as "structure documents. . . in which the objects. . . can be acted upon by, and can themselves act upon, other objects in the document or the outside world" (p. 78), and (2) that clicking on a face plays an audio name, etc. (p. 79). These portions are irrelevant to the claims, which have to do with use of objects in connection with transformation of a print data stream, as discussed above.

75. Regarding claims 5, 26, and 42, Samsung cited Interleaf's reference to LISP (e.g., Interleaf, pp. 77-78). However, nothing suggests extracting the data at print time, that is, based on an input print data stream, much less doing so using the AS/400's DDS.

76. Regarding claims 6, 27, and 43, the reference to Interleaf's documents "not appear[ing] active to all users" further reinforces the point that Interleaf relates to user editing and handling of documents, not their printing. The combination of IBM and Interleaf does not disclose that the script automatically receiving data also requests this data automatically in the context of a print data stream.

77. Regarding claims 7, 28, and 44, IBM does not disclose scripts, but merely the use of a compiled DDS on data. Accordingly, the functionality of the scripts recited in these claims cannot be performed with compiled DDS. Even if Interleaf discloses such functionality, it would not have been possible to incorporate this into the AS/400, as suggested by Samsung.

78. This argument applies, as well, to claims 11, 32, and 48.

79. Regarding claims 12, 33, and 49, the cited features of Interleaf operate while the document is being edited or opened; they do not relate to manipulation of a print stream produced by an Interleaf document. In any event, it is far from clear that it would have been possible to perform this functionality in the DDS used by the AS/400 system.

III. <u>Ground of Rejection No. 3: IBM in view of Interleaf and Interleaf Patent</u>

80. The Interleaf Patent does not add relevant subject matter to Interleaf. Both deal with document creation and editing. At most, the software described in the Interleaf Patent may generate a print stream, but it does not interpret or transform a print stream, as claimed in the '789 patent.

81. Regarding claims 2, 23, and 39, the Interleaf Patent, at col. 2, line 61 to col. 3, line 13, bears no relation to transforming a print data stream, as claimed in the '789 patent.

82. Regarding claims 4, 25, and 41, the cited portion of the Interleaf Patent relates to a user-document interface, not a printing function. It is not clear at all how this relates to printing, or how the AS/400's DDS would have been capable of providing this functionality.

83. Regarding claims 5, 26, and 42, the cited portions of the Interleaf patent do not relate to use in connection with an input print data stream. For example, one application involves retrieving stock information and displaying it in text and charts; however, nothing suggests extracting the data at print time, that is, based on an input print data stream, much less doing so using the AS/400's DDS.

84. Regarding claims 6, 27, and 43, Interleaf and the Interleaf Patent relate to user editing and handling of documents, not their printing. Moreover, nothing suggests using the AS/400's DDS, to provide the functionality described in the Interleaf Patent, or that it is even possible to do so.

85. Regarding claims 7, 28, and 44, IBM does not disclose scripts, but merely the use of a compiled DDS on data. The portion of the Interleaf Patent that discloses a "customer receipt that

automatically runs a 'frame grabber' and incorporates a photographic image of the purchaser" does not transpire based on an input print data stream, but rather during creation of a document. Nor is there any indication that the DDS used in the AS/400 would have been capable of performing such functionality.

86. Regarding claims 8, 29, and 45, the cited portions of the Interleaf Patent have no relevance to a print application. Thus, for example, there is no reason to think that one of ordinary skill would have combined the references to produce an urgent memo that when printed integrates with voice annotation software so when it prints itself, it announces its presence audibly, or that this would have been possible using the tools available from IBM.

87. Regarding claims 9, 30, and 46, the disclosure of the Interleaf Patent for a "WYSIWYG document that can send itself over an electronic mail system" has no application to a transformation of a print data stream, as recited in the claims.

88. Regarding claims 10, 31, and 47, the Interleaf Patent's example of a "stock-market information center that retrieves stock information and displays it in text and charts that can be automatically distributed or incorporated in other documents" is in the limited context of document creation and editing. Once the document is sent to the printer, that functionality is lost. Nothing in the references suggests obtaining such information after receiving an input print data stream.

89. Regarding claims 11, 32, and 48, nothing suggests combining the features of the Interleaf Patent at col. 6, lines 6-16, and col. 3, lines 37-39 with IBM, which does not disclose scripts, but merely the use of a compiled DDS on data. Moreover, even if the Interleaf Patent discloses the functionality of these claims, it would not have been possible to incorporate this into the AS/400.

90. Regarding claims 12, 33, and 49, the portion of the Interleaf Patent that discloses a "document created from a database which updates the database if you make any changes in the imported document" has no application to a transformation of a print data stream, as recited in the claims, nor does it seem possible to perform this functionality in the AS/400.

91. Regarding claims 13, 14, 34, 35, 50, and 51 the cited examples are not relevant to the inventions of the '789 patent. First, the examples relate to the generation of a document, not an operation performed on a print stream based on a document. Second, simply comparing an announcement date to a current system date is not a timer function. It is not at all clear it would have been possible to implement either the date comparison or the timer functions on the AS/400.

92. Regarding claims 16, 37, and 53, the various editing functions disclosed in the Interleaf Patent are not relevant to the '789 patent, because they are performed before generating an

input print data stream. Nowhere does the Interleaf Patent disclose or render obvious that these functions are performed after receiving an input print data stream, as required by the claims.

93. Regarding claim 18, the cited functions of the Interleaf Patent occur before the objects are output in the output print data stream. The Interleaf Patent does not disclose performing them after receiving a print data stream.

IV. Ground of Rejection No. 4: IBM in view of Interleaf/Interleaf Patent and Lieberman

94. Lieberman relates to agents to interact with multiple (existing) user applications.

95. The field of the invention of the '789 patent is printer software generally, and in particular, interpretation or processing of print data streams at a printer or print server.

96. Lieberman does not relate to printing, or to interpretation of print data streams at a printer or print server. It has nothing to do with the field of the invention, printing, print streams, printers, or print servers. There would have been no reason for one of ordinary skill to refer to Lieberman, or to use its teachings to solve a problem relating to interpretation of print data streams at a printer or print server.

V. Ground of Rejection No. 5: Interleaf

97. Generally, the objective of the Interleaf system is to build an enhanced user interface (UI). Interleaf is a "document creation component" (Interleaf, p. 75).

98. In contrast, the '789 patent deals with printing, which may include, for example, the output of user applications such as the Interleaf document creation system. The '789 patent addresses an entirely different part of the workflow, namely, the printing stage. That is, the patented technology takes over where the Interleaf system ends.

99. There are at least four fundamental differences between Interleaf and the '789 patent. Interleaf does not: disclose a method for the transformation of print data streams; read in an input print stream; parse an input print data stream; or produce an output print data stream based on a transformed input data stream.

A. No Transformation of Print Data Streams

100. The '789 patent claims a method, computer-readable medium, and computer signal for the transformation of digital print data streams. Statements at Interleaf, p. 76 and 81-84, refer to editing a working document in the user interface, not printing it.

B. No Reading in of an Input Print Stream

101. The claims recite that an input print data stream is read in. Interleaf p. 84 ("If a document contains active objects within the document file stream. . . these become activated when the document is opened programmatically or by the user for viewing, editing, or printing") refers to a

document file stream, not a print data stream. The document file stream is merely what is read in by the Interleaf system when the Interleaf document is opened. It is not a print data stream, as recited in the claims.

102. Opening an Interleaf document is akin to opening a Word document. However, this operation does not create a print data stream. In the Word analogy, print data are only generated when the print function is activated, e.g., a print icon in Word is clicked.

103. In order to print a document, Interleaf may conceivably generate a print data stream as its output, but it does not read in a print data stream as its input, as claimed in the '789 patent.

C. No Parsing of an Input Print Data Stream

104. The claims recite that the input print data stream are analyzed by a parser for graphically representable objects. The Lisp interpreter operating on Interleaf does not operate on a print data stream. Thus, even assuming the Lisp interpreter were a parser, it does not parse a print data stream.

105. In addition, the Lisp interpreter (by definition) only interprets LISP scripts, and nothing else. Therefore, it is incapable of interpreting print data streams, as contrasted to the parser of the claims of the '789 patent, which analyzes an input print data stream.

D. No Output Print Data Stream Based on a Transformed Input Print Data Stream

106. Although Interleaf, like other document creation tools, may eventually create a print data stream it does not read that stream and analyze it using a parser and split it up into objects. Interleaf cannot transform objects and combine those transformed objects into an output print data stream.

107. Interleaf's "[o]pening the document for printing" does not: (1) read the print data stream; (2) parse it; (3) split it up into objects; (4) transform objects to control a printer; or (5) combine those transformed objects into an output data stream.

E. Dependent Claims

108. For claims 2, 23, and 39, Interleaf's disclosure of a template document subclass with an 'open' method used for auto-localizing documents bears no relation to transforming a print data stream.

109. Portions in Interleaf such as: (1) that active documents are defined as "structure documents. . . in which the objects. . . can be acted upon by, and can themselves act upon, other objects in the document or the outside world." and (2) the Intelligent Maintenance Aid (IMA) application example, which "make use of the system's printing and filtering capabilities" are

irrelevant to claims 4, 25, and 41, which have to do with use of objects in connection with transformation of a print data stream, as discussed above.

110. Regarding claims 5, 26, and 42, Interleaf's reference to LISP (e.g., pp. 77-78) does not suggest extracting the data at print time, that is, based on an input print data stream.

111. Regarding claims 6, 27, and 43, Interleaf's documents "not appear[ing] active to all users" further shows that Interleaf relates to user editing and handling of documents, not their printing.

112. Interleaf did not anticipate claims 7, 28, and 44, at least for the reason that it does not disclose every element of the independent claims from which these claims depend.

113. Interleaf did not anticipate claims 11, 32, and 48, at least for the reason that it does not disclose every element of the independent claims from which these claims depend.

114. Regarding claims 12, 33, and 49, the cited features of Interleaf operate while the document is being edited or opened; they do not relate to manipulation of a print stream produced by an Interleaf document.

VI. <u>Ground of Rejection No. 6: Interleaf in view of Lieberman</u>

115. As discussed above, with reference to Ground of Rejection No. 4, Lieberman does not involve printing. A person of ordinary skill in the relevant art would have had no reason to combine IBM or Interleaf with Lieberman.

116. Moreover, for reasons explained above, even the combined Lieberman and Interleaf would not result in the claimed inventions.

VII. <u>Grounds of Rejection No. 7: Interleaf in view of Interleaf Patent</u>

117. As discussed above, the Interleaf Patent, like Interleaf, does not relate to transforming a print data stream.

118. For example, the Interleaf Patent includes the following sentence: "As explained above, any data object within the hierarchy may be associated with a method which, upon the occurrence of a certain event, will be implemented." (Col. 3, lines 37-39). However, read in context, it is clear that the occurrences upon which methods are implemented are in the context of user interactions with the document, e.g., user opening a document, clicking on an image. They are not in a print data stream, nor are they implemented in the context of printing a document, as claimed in the '789 patent.

119. Regarding claims 2, 23, and 39, neither Interleaf nor the Interleaf Patent bears any relation to transforming a print data stream.

120. Regarding claims 4, 25, and 41, as with the example provided by a "phone directory that can call the person who's [sic] name you've selected; if the line is busy, it puts you into electronic mail," the cited portions of the Interleaf documents relate to a user-document interface, not a printing function.

121. Regarding claims 5, 26, and 42 and claims 6, 27, and 43, the cited portions of Interleaf and the Patent do not relate to use in connection with an input print data stream. For example, one application involves retrieving stock information and displaying it in text and charts. However, this capability is discussed with respect to the limited scope of document creation and editing. Once the document is sent to the printer, that functionality is lost. Nothing in the references suggests obtaining such information during the printing process, i.e., after receiving an input print data stream.

122. Regarding claims 7, 28, and 44, the portion of the Interleaf Patent that discloses a "customer receipt that automatically runs a 'frame grabber' and incorporates a photographic image of the purchaser" does not transpire based on an input print data stream, but during creation of a document.

123. Regarding claims 8, 29, and 45, the portions of the Interleaf Patent, including an "urgent memo that integrates with voice annotation software so when it mails itself to someone, it announces its presence audibly" and a "document created from a database which updates the database if you make any changes in the imported document" have no application in a print application.

124. Regarding claims 9, 30, and 46, the Interleaf Patent's cited disclosure has no application to a transformation of a print data stream, as recited in the claims.

125. Regarding claims 10, 31, and 47, the Interleaf Patent's example of a "stock-market information center that retrieves stock information and displays it in text and charts that can be automatically distributed or incorporated in other documents" does not suggest obtaining such information after receiving an input print data stream.

126. Regarding claims 11, 32, and 48, the cited Interleaf Patent at col. 6, lines 6-16, and col. 3, lines 37-39, does not disclose or otherwise relate to transformation of a print data stream.

127. Regarding claims 12-14, 33-35, and 49-51, the cited portion of the Interleaf Patent has no application to a transformation of a print data stream, as recited in the '789 patent claims. In addition, the examples, e.g., a document that includes "Draft" before an announcement date, a reminder that appears on a screen based on a document due date, relate to the generation of a

document, not an operation performed on a print stream based on a document. They are not relevant to the claims.

128. Regarding claims 16, 37, and 53, the various editing functions disclosed in the Interleaf Patent are performed before generating an input print data stream. Nowhere does the Patent disclose or render obvious that these functions are performed after receiving an input print data stream, as required by the claims.

129. Regarding claim 18, the cited functions of the Interleaf Patent occur before the objects are output in the output print data stream. The Patent does not disclose performing them after receiving a print data stream.

VIII. <u>Ground of Rejection No. 8: Interleaf in view the Interleaf Patent and Lieberman</u>

130. As discussed above, with reference to Ground of Rejection No. 4, Lieberman does not involve printing. A person of ordinary skill in the relevant art would have had no reason to combine IBM or the Interleaf references with Lieberman for any reason.

IX. <u>Ground of Rejection No. 9: Interleaf Patent in view of IBM</u>

131. The above remarks pertaining to the combination of the Interleaf Patent and IBM are incorporated herein by reference, and are not repeated here.

132. In addition to the above, there is no overlap between the Interleaf Patent and IBM. The print data stream produced by the Interleaf Patent as an output is handed off to IBM as an input. For reasons explained above in paragraph 64, this combination would not result in any claimed invention.

133. Furthermore, reference to various types of print data streams is irrelevant, because if a PC running the Interleaf software were operated on a PC linked to an AS/400, these would simply be converted to AFP data streams for processing.

I declare under penalty of perjury that all statements made here are based on my own knowledge and are believed to be true. I understand that willful false statements and the like are punishable by fine or imprisonment, or both (18 U.S.C. Section 1001), and may jeopardize the validity of the patent subject to reexamination.

David B

March 30, 2011

David Birnbaum, Ph.D. Date

Exhibit A

David Birnbaum
2114 Sand Dollar Dr
Richmond, CA 94804
db272@cornell.edu

EXPERIENCE

University of North Carolina – Greensboro **2003- 2007**
Adjunct Professor of Physics

Xerox Corporation **1980-2003**
Engineering Fellow and Principal Scientist

General Railway Signal **1976-1980**
Senior Research Engineer and Advisory Research Engineer

Hamline University **1971-1976**
Assistant Professor (Physics)

Carnegie-Mellon University **1967-1971**
Research Associate and Assistant Professor (Physics)

University of Rochester **1967**
Research Associate (Physics)

EDUCATION

M.B.A., Executive Development Program. Graduated with Honors. 1980-1981
University of Rochester Rochester, NY

M.A, Ph.D., Physics. Thesis in experimental elementary particle physics. 1961-1967
University of Rochester Rochester, NY

B. Engr., Engineering Physics. 1956-1961
Cornell University Ithaca, NY

HONORS and AFFILIATIONS

Xerox President's Award
Life Senior Member, Institute of Electrical and Electronic Engineers
Member, American Physical Society
Member, Sigma Xi, Scientific Honorary Society
Member, Beta Gamma Sigma, Business Honorary Society

SKILLS

Registered Professional Engineer (NY)
Registered Patent Agent USPTO
Over 40 years of computing experience including facility in: FORTRAN, C, C++, Unix and Linux
Amateur Radio Extra Class License

PATENTS & PUBLICATIONS

15 US Patents
12 papers in referred scientific journals

Exhibit B

IBM

eserver

iSeries

Printer Device Programming

Version 5

SC41-5713-05

IBM

eserver®

iSeries

Printer Device Programming

Version 5

SC41-5713-05

Note

Before using this information and the product it supports, be sure to read the information in "Notices" on page 539.

Sixth Edition (September 2002)

This edition replaces SC41-5713-04. This edition applies only to reduced instruction set computer (RISC) systems.

Contents

Chapter 3. Spool support 123

Chapter 4. Remote System Printing 159

Chapter 5. Working with the OS/400 Network Print Server 187

Part 3. Advanced function printing (AFP) 195

Chapter 6. What Is Advanced Function Printing (AFP)? 197

Chapter 7. Resources needed to perform Advanced Function Printing (AFP) 201

The printer file open operation is controlled by parameters specified in the printer file, the high-level language program, and in printer file overrides (through the OVRPRTF command). See "Overriding printer files" on page 69 for more information on overrides.

As an example, if the printer file specified lines per inch (LPI) of 8, and an OVRPRTF command specified an LPI of 6, the LPI of 6 would be used since the override value specified by the OVRPRTF command takes precedence over the LPI value specified in the printer file.

Output processing

Part **2** of the application program performs the operations of reading, compiling, and sending the output to the output queue specified in the OUTQ parameter of the CRTPRTF command, or to the printer specified in the DEV parameter of the CRTPRTF command. In this example, the SPOOL parameter has a value of (*YES) which means the output will become a spooled file in the designated output queue.

Unless otherwise noted, the printer file parameters listed in Output processing on page 56 are also valid for externally-described printer files.The following printer file parameters from the CRTPRTF command are additional parameters that are looked at by the application program during the output processing.

SRCFILE
Specifies the qualified name of the source file and member, if one exists, that contains the data description specifications (DDS).

Since this example uses DDS, look at **1** in the program listing and see that the name of the printer file is LABELPR3. LABELPR3 was compiled using the source from the member and file that are listed here. See "DDS keywords" on page 62 for an example of the compiled DDS and a list of DDS keywords.

Note: The DDS will be compiled before the application program runs. The application program never looks at the DDS file and member, only at the compiled results.

Option
Specifies the type of printout that will be produced when the printer file is created.

GENLVL
Specifies the severity level of DDS messages that cause file creation to fail.

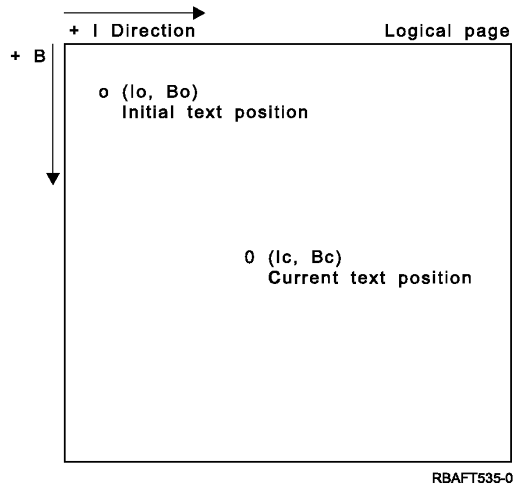
Data description specifications

Below is the example of the compiled DDS used by the RPG program. You can update the DDS; however, you must then re-compile it.

```
000100900115            R HEADNG
000200900115                                    3  2'MAILING LABELS'
000300900115
000400900115            R DETAIL1
000500900115              NAME          25      2  2UNDERLINE
000600900115              ADD1          25      3  2
000700900115            R DETAIL3
000800900115              ADD2          25         2SPACEB(1)
000900900115            R DETAIL4
001000900115              CTSTZP        30         2HIGHLIGHT SPACEB(1)
```

This example uses three DDS keywords: SPACEB, UNDERLINE, and HIGHLIGHT.

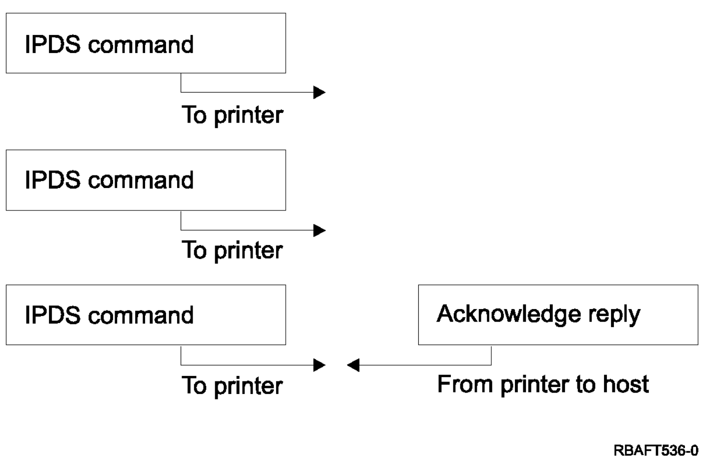
text coordinate (Ic) and the current baseline text coordinate (Bc).



Processing IPDS Commands

The structured field format of IPDS allows commands to be sent to the printer in a continuous stream. Each command is self-describing. The command length, identifier, flag byte, and data (not always present) are all part of each command. The printer-host conversation is carried on as if IPDS commands were processed in sequential order by the printer.

Every IPDS command contains a flag byte. The setting on the acknowledgement-required bit on this flag byte indicates the end of a command sequence to the printer. The printer then sends an acknowledge reply to the host, as illustrated in the following diagram:



The IPDS Command Format

All IPDS commands are encoded in the following format:

Length	Command	Flag	CID	Data

Length

A 2-byte field that specifies the length of the command. This count

includes itself, the command field, the flag byte and the optional correlation ID (CID), and data fields. The length field can range from X'0005' to X'7FFF'.

Command
A 2-byte field that specifies the IPDS command.

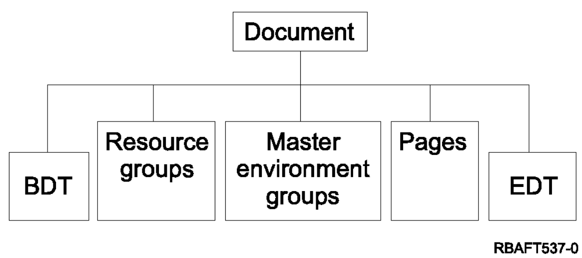
Flag A 1-byte field that contains the IPDS command stream flags.
- Bit 0 is the acknowledgement required (ARQ) flag. If this bit is on, the host requests the printer to send an acknowledge reply.
- Bit 1 is the correlation ID (CID) flag. If it is on, a 2-byte correlation ID follows. If it is off, the CID is not present and the following bytes (if any) contain the data field.

CID (correlation ID)
A 2-byte field that specifies an identifier for the command. A presentation services program can use any value between X'0000' and X'FFFF' for the correlation ID.

Data Not present for all commands. If present, it contains specific orders, parameters, and data appropriate for the given command.

IPDS Operating States

IPDS commands are defined within the context of printer operating states. The printer moves between these operating states during command processing. IPDS printers are *state machines* with the following operating states:



- Home state
- Block state
  - IO image block state
  - IM image block state
  - Graphics block state
  - Bar code block state.
- Page state
- Overlay state
- Page segment state
- Font state
- Any-state

Home state
The initial IPDS operating state. The printer returns to home state at the end of each downloaded page, page segment, coded font, or overlay.

While in home state, the printer receives control and initialization commands to prepare for the print operation. In home state, the printer

can also receive commands that delete resources or request the return of printer information to the host presentation services program.

Block states
State for establishing the initial processing conditions for a block of data and placing the block of data on the logical page, page segment or overlay. The printer can only enter a block state from page, page segment, or overlay states.

Page state
The operating state for printing a logical page. The printer enters page state from home state on receiving a Begin Page command and exits on receiving an End Page command.

In page state, the printer can receive commands that merge previously defined and loaded overlays and page segments with the current page information. The printer can also receive Write Text commands that position text on the logical page and can enter a block state to write image, bar code, and graphics blocks.

Overlay state
State that allows overlay data to be stored in the printer. The printer enters overlay state from home state on receiving a Begin Overlay command and exits on receiving an End Page command.

In overlay state, the printer can receive commands that merge previously defined and loaded overlays and page segments with the current page information. The printer can also receive Write Text commands that position text on the logical page and can enter a block state to write image, bar code, and graphics blocks.

Page segment state
State that allows page segment data to be stored in the printer. The printer enters page segment state from home state on receiving a Begin Page Segment command and exits on an End Page command.

In page segment state, the printer can receive Write Text commands that position text on the logical page and can enter a block state to write image, bar code, and graphics blocks.

Font state
State that allows the printer to receive downloaded coded-font data. The printer enters font state from home state on receiving a Load Font Control command.

While the printer is in font state, the Load Font command can send coded-font, character-raster pattern data to the printer. Receipt of an End command returns the printer to home state.

Any-state
Some IPDS commands can be received in any IPDS operating state. These commands do not change the IPDS operating state, with the exception of XOA Discard Buffered Data.

Default Handling

Defaults are values used as control parameters when no other values are specified in the current command. IPDS defaults are invoked through omission or through values transmitted in the data field portion of commands. The IPDS default structure is normally hierarchical. General IPDS default rules are:

- If power has been interrupted or if the printer has been initialized, printer-established page default values are used until specific IPDS default values are received.
- Initial page values are established when the printer receives a Load Page Descriptor command. If no such command is received, printer-established default values remain in effect.
- Initial data block values are established when the printer receives either a Write Image Control, Write Image Control 2, Write Bar Code Control, or Write Graphics Control command. These values remain in effect until data controls override them or until the printer receives an End command that ends the block.

Mixed Object: Document Content Architecture (MO:DCA)

The ability to print documents with consistent output, independent of either operating system or printer, is extremely important to the user of printed data. In order to help achieve this goal, IBM has defined a single object-oriented data stream—**Mixed Object Document Content Architecture (MO:DCA)**. (An object is a collection of data that can be treated as a unit.) This architecture has been developed in order to meet several objectives:

- The requirements relating to document and data sharing specified in IBM's Systems Application Architecture
- Co-existence and migration of existing IBM document architecture and printer data streams
- Device independence
- Separation of functions to simplify transformation of objects into other data streams
- National Language Support
- Office Document Architecture (ODA) support
- Standard Generalized Markup Language (SGML)

The strategic architecture for the interchange of revisable and presentation form of documents and objects used as resources is MO:DCA, which has evolved from Revisable Form Text: Document Content Architecture (RFT:DCA).

The data stream for an MO:DCA document consists of various objects, such as text, images, and graphics, as well as the logical and layout structure of the document. The logical structure defines the logical content of the document—chapters, figures, and lists. The layout structure defines the way the data should be presented.

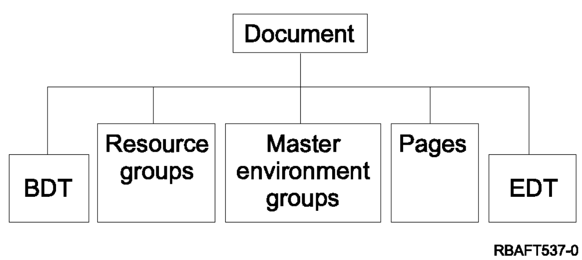


Figure 8. MO:DCA Document Structure

BDT (Begin document)
Indicates the beginning of the document

Resource groups
Specifies fonts, overlays, and segments so that these objects can be transmitted as part of the data stream. They can be referenced by an MO:DCA Include structured field.

Master environment groups
Specifies the processing environment, such as space definitions, suppression of data, number of copies, and internal data stream references.

Pages Contains objects that are part of the document. These objects could be text, graphics, and images.

EDT (End document)
Indicates the end of the document.

The following different types of objects make up MO:DCA. All of these objects are supported by IPDS:

- Bar Code Object Content Architecture (BCOCA)
- Image Object Content Architecture (IOCA)
- Graphics Object Content Architecture (GOCA)
- Presentation Text Object Content Architecture (PTOCA)
- Font Object Content Architecture (FOCA).

Bar Code Object Content Architecture (BCOCA)

A bar-code object could contain “draw rule” commands or raster data, depending on whether the bar code is to be drawn as a graphics object or has been scanned into the data stream as an image. A bar code object containing draw rule commands is built up using only lines of a specified length and width. A graphics object is constructed from a number of primitives, such as lines, arcs, symbols, shaded areas, and point arrays.

Image Object Content Architecture (IOCA)

IOCA represents images in device-independent format. A standard set of constructs has been defined to describe the image data, the characteristics of that data, and manipulation functions that may be performed on the data. The image content is inserted in an image segment.

Graphics Object Content Architecture (GOCA)

GOCA describes complex pictures. These pictures are formed from a collection of primitives, such as lines, arcs, characters, symbols, and shaded areas or point arrays. Each of these primitives has its own set of attributes, such as line width, orientation, and resolution. In addition to these attributes, there is a set of general drawing attributes like color, which apply to all primitives.

Presentation Text Object Content Architecture (PTOCA)

PTOCA describes the text part of a document. The presentation text object, in common with the other objects, is designed not only to be carried by, but to be an integral part of, the data stream, providing the following:

- Structured field introducer and syntax for the structured field
- Begin/end object structure
- Control of alternate action selection for error recovery
- Passing of exception conditions back to the originating process
- Initial state of the object
- Relationship of presentation text objects to other objects contained in the data stream.

Two structured fields provide the necessary presentation information to the printer:

P T descriptor structured field
Defines several positional parameters for the object

P T data structured field
Contains the presentation text and the control sequences for positioning graphic characters. These graphic characters are defined within the coded fonts.

Font Object Content Architecture (FOCA)

In order to achieve uniform document presentation output, it is essential that font resources be consistently defined and implemented. These resources must be identified by means of a constant, unvarying set of parameters.

FOCA makes it possible to achieve the required degree of consistency by defining:

- A common font and character definition model that can be used by all products and architectures as the basis for font applications
- A composite set of parameters specific to a font resource and references to that resource
- A device-and-technology-independent method of defining font measurements
- The specification of formats for conveying font information to suit the application

FOCA defines the parameter content of:

- IBM font resources
- References to the font resources
- Information accessed by the font resources

American National Standard Code for Information Interchange (ASCII)

There is no formal structure controlling the use of the ASCII data stream to control printers attached to systems providing ASCII support. Control of page printers, like the IBM 3812, is exercised using page map primitives (PMPs), which are a set of commands or basic instruction set of these printers when attached in ASCII mode. ASCII data sent to a page printer is translated into PMPs. The page printer composes the page of data in its internal memory or page map. Two page orientations (portrait and landscape) as well as four print directions are supported. Complexity of the printed data is determined by the application print program, which can set the pels on explicitly in the page set, or implicitly, by instructing the printer to generate characters or vectors (lines). Fonts available for printing are stored on the printer's microcode/font diskette. Most page printers support **macros**, which are a saved list of PMP commands, avoiding the necessity for the application program to send down a string of individual commands each time a particular printed function is required.

There are five basic categories of PMP commands:

Page commands
Set overall page parameters, such as size and orientation

Cursor commands
Move the cursor on the page map

Font commands
Manage fonts within the page printer

Generation commands
Create pels on the page map

Macro commands
Allow strings of other commands to be saved for later processing.

Printing capabilities and functions in ASCII attach mode are governed by individual application programs that are written to suit the capabilities of specific printers (or printers that provide an emulation of that printer). There is no architectural data stream standard to which ASCII printers can conform in the interests of uniformity. ASCII printing applications are therefore totally printer dependent.

On OS/400, ASCII printing support is provided by translating native EBCDIC characters to the ASCII equivalents.

PostScript

PostScript is not a data stream but rather a page description language developed by Adobe Systems Incorporated and used to prepare page layouts for printing. This involves not only text with typesetting information on sizes and fonts, but graphics and scanned images as well. An example is Aldus Pagemaker**, which produces PostScript for downloading to the IBM personal pageprinter adapter. This adapter contains the PostScript Interpreter developed for IBM by Adobe Systems Incorporated. The Interpreter produces the video image for the IBM Personal Pageprinter 4216 Model 20, 4019, or 4029 with the postscript feature.

The following licensed programs produce PostScript output:
- IBM Personal Publishing System
- IBM Interleaf Publishing Series
- Document Composition Facility (DCF) Release 3.2
- Image Handling Facility (IHF) Release 1.2.

There is currently no support for PostScript on the iSeries server.

Exhibit C

AS/400e

Printer Device Programming

Version 4

SC41-5713-03

Note

Before using this information and the product it supports, be sure to read the information in "Notices" on page 605.

Fourth Edition (May 1999)

This edition applies to version 4 release 3, modification 0 of the licensed program IBM Operating System/400 (Program 5769-SS1), and to all subsequent releases and modifications until otherwise indicated in new editions. This edition applies only to reduced instruction set computer (RISC) systems.

This edition replaces SC41-5713-02.

Contents

A printer file is opened to prepare the system so that the application can put data into a spooled file or print it out directly to a printer. Information from the high-level language application program, the printer file, and any printer file overrides is combined.

The printer file open operation is controlled by parameters specified in the printer file, the high-level language program, and in printer file overrides (through the OVRPRTF command). See "Overriding Printer Files" on page 69 for more information on overrides.

As an example, if the printer file specified lines per inch (LPI) of 8, and an OVRPRTF command specified an LPI of 6, the LPI of 6 would be used since the override value specified by the OVRPRTF command takes precedence over the LPI value specified in the printer file.

Output Processing

Part **2** of the application program performs the operations of reading, compiling, and sending the output to the output queue specified in the OUTQ parameter of the CRTPRTF command, or to the printer specified in the DEV parameter of the CRTPRTF command. In this example, the SPOOL parameter has a value of (*YES) which means the output will become a spooled file in the designated output queue.

Unless otherwise noted, the printer file parameters listed in "Output Processing" on page 56 are also valid for externally-described printer files.The following printer file parameters from the CRTPRTF command are additional parameters that are looked at by the application program during the output processing.

SRCFILE
Specifies the qualified name of the source file and member, if one exists, that contains the data description specifications (DDS).

Since this example uses DDS, look at **1** in the program listing and see that the name of the printer file is LABELPR3. LABELPR3 was compiled using the source from the member and file that are listed here. See "Data Description Specifications" for the example of the compiled DDS and a list of DDS keywords.

Note: The DDS will be compiled before the application program runs. The application program never looks at the DDS file and member, only at the compiled results.

Option
Specifies the type of printout that will be produced when the printer file is created.

GENLVL
Specifies the severity level of DDS messages that cause file creation to fail.

Data Description Specifications

Below is the example of the compiled DDS used by the RPG program. You can update the DDS; however, you must then re-compile it.

```
000100900115                R HEADNG
000200900115                                                3  2'MAILING LABELS'
000300900115
```

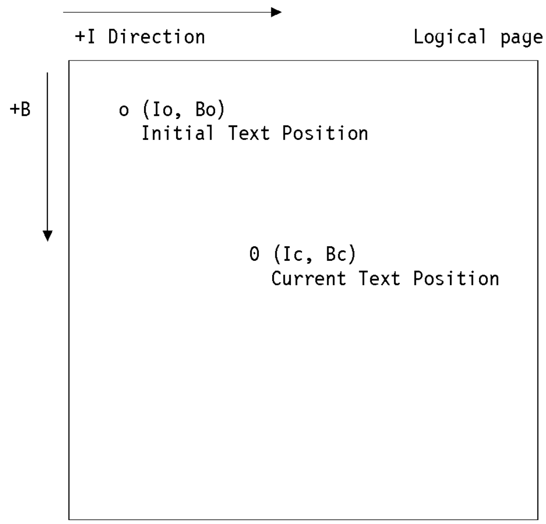


Figure 21. I,B Coordinate System on the Logical Page

Processing IPDS Commands

The structured field format of IPDS allows commands to be sent to the printer in a continuous stream. Each command is self-describing. The command length, identifier, flag byte, and data (not always present) are all part of each command. The printer-host conversation is carried on as if IPDS commands were processed in sequential order by the printer.

Every IPDS command contains a flag byte. The setting on the acknowledgement-required bit on this flag byte indicates the end of a command sequence to the printer. The printer then sends an acknowledge reply to the host, as illustrated in the following diagram:

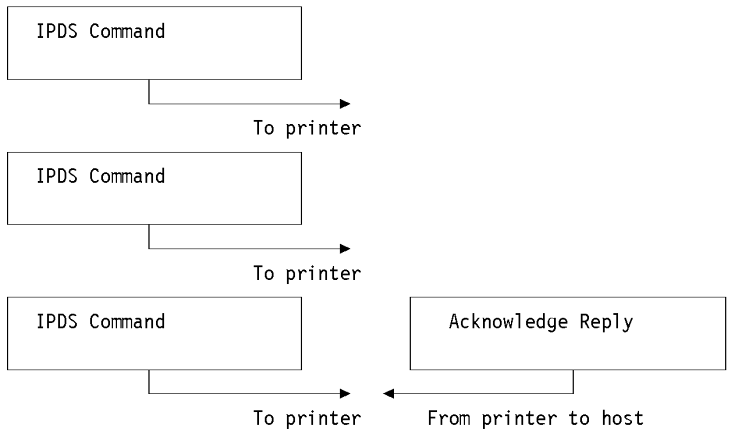


Figure 22. Example of IPDS Command Processing

The IPDS Command Format

All IPDS commands are encoded in the following format:

Length	Command	Flag	CID	Data

Length
A 2-byte field that specifies the length of the command. This count includes itself, the command field, the flag byte and the optional correlation ID (CID), and data fields. The length field can range from X'0005' to X'7FFF'.

Command
A 2-byte field that specifies the IPDS command.

Flag A 1-byte field that contains the IPDS command stream flags.

- Bit 0 is the acknowledgement required (ARQ) flag. If this bit is on, the host requests the printer to send an acknowledge reply.
- Bit 1 is the correlation ID (CID) flag. If it is on, a 2-byte correlation ID follows. If it is off, the CID is not present and the following bytes (if any) contain the data field.

CID (correlation ID)
A 2-byte field that specifies an identifier for the command. A presentation services program can use any value between X'0000' and X'FFFF' for the correlation ID.

Data Not present for all commands. If present, it contains specific orders, parameters, and data appropriate for the given command.

IPDS Operating States

IPDS commands are defined within the context of printer operating states. The printer moves between these operating states during command processing. IPDS printers are *state machines* with the following operating states:

- Home state
- Block state
  - IO image block state
  - IM image block state
  - Graphics block state
  - Bar code block state.
- Page state
- Overlay state
- Page segment state
- Font state
- Any-state

Home state
: The initial IPDS operating state. The printer returns to home state at the end of each downloaded page, page segment, coded font, or overlay.

While in home state, the printer receives control and initialization commands to prepare for the print operation. In home state, the printer can also receive commands that delete resources or request the return of printer information to the host presentation services program.

Block states
: State for establishing the initial processing conditions for a block of data and placing the block of data on the logical page, page segment or overlay. The printer can only enter a block state from page, page segment, or overlay states.

Page state
: The operating state for printing a logical page. The printer enters page state from home state on receiving a Begin Page command and exits on receiving an End Page command.

In page state, the printer can receive commands that merge previously defined and loaded overlays and page segments with the current page information. The printer can also receive Write Text commands that position text on the logical page and can enter a block state to write image, bar code, and graphics blocks.

Overlay state
: State that allows overlay data to be stored in the printer. The printer enters overlay state from home state on receiving a Begin Overlay command and exits on receiving an End Page command.

In overlay state, the printer can receive commands that merge previously defined and loaded overlays and page segments with the current page information. The printer can also receive Write Text commands that position text on the logical page and can enter a block state to write image, bar code, and graphics blocks.

Page segment state
: State that allows page segment data to be stored in the printer. The printer

enters page segment state from home state on receiving a Begin Page Segment command and exits on an End Page command.

In page segment state, the printer can receive Write Text commands that position text on the logical page and can enter a block state to write image, bar code, and graphics blocks.

Font state
State that allows the printer to receive downloaded coded-font data. The printer enters font state from home state on receiving a Load Font Control command.

While the printer is in font state, the Load Font command can send coded-font, character-raster pattern data to the printer. Receipt of an End command returns the printer to home state.

Any-state
Some IPDS commands can be received in any IPDS operating state. These commands do not change the IPDS operating state, with the exception of XOA Discard Buffered Data.

Default Handling

Defaults are values used as control parameters when no other values are specified in the current command. IPDS defaults are invoked through omission or through values transmitted in the data field portion of commands. The IPDS default structure is normally hierarchical. General IPDS default rules are:

- If power has been interrupted or if the printer has been initialized, printer-established page default values are used until specific IPDS default values are received.
- Initial page values are established when the printer receives a Load Page Descriptor command. If no such command is received, printer-established default values remain in effect.
- Initial data block values are established when the printer receives either a Write Image Control, Write Image Control 2, Write Bar Code Control, or Write Graphics Control command. These values remain in effect until data controls override them or until the printer receives an End command that ends the block.

Mixed Object: Document Content Architecture (MO:DCA)

The ability to print documents with consistent output, independent of either operating system or printer, is extremely important to the user of printed data. In order to help achieve this goal, IBM has defined a single object-oriented data stream—**Mixed Object Document Content Architecture (MO:DCA)**. (An object is a collection of data that can be treated as a unit.) This architecture has been developed in order to meet several objectives:

- The requirements relating to document and data sharing specified in IBM's Systems Application Architecture
- Co-existence and migration of existing IBM document architecture and printer data streams
- Device independence
- Separation of functions to simplify transformation of objects into other data streams
- National Language Support
- Office Document Architecture (ODA) support

Exhibit D

IBM Terminology

- A
- B
- C
- D
- E
- F
- G
- H
- I
- J
- K
- L
- M
- N
- O
- P
- Q
- R
- S
- T
- U
- V
- W
- X
- Y
- Z
- #

This site contains terms and definitions from many IBM software and hardware products as well as general computing terms.

S

S/390
IBM enterprise servers based on Enterprise Systems Architecture/390 (ESA/390).The S/390 has been superseded by the IBM zSeries.

S/390 storage
Storage arrays and logical volumes (LVOLs) that are connected to S/390 servers. S/390 storage sometimes also includes zSeries storage. See also zSeries storage.

SA
(1) See system administrator.
(2) See Security Association.

screen-image interface
The part of the Front End Programming Interface that has a buffer with one byte for each screen position.

screen import
The process of importing a screen definition (in its current state) and saving it to a screen file within the 3270 terminal service tools workbench, for the purpose of generating recognition profiles and custom screen records. Use the 3270 terminal service recorder to import screens.

screen page
The amount of data displayed, or capable of being displayed, at any one time on the screen of a terminal.

screen reader
A device that renders onscreen text as audible language. See also digital speech synthesizer.

screen recognition
A runtime function that determines the state of a screen and processes the screen in accordance with the identifiers in the recognition profiles. Screen recognition compares the screen as presented by the 3270 application to the defined recognition profiles to determine which screen state applies.

screen recognition criteria
A set of criteria used to determine whether a host screen matches a screen customization and should have that screen customization's actions applied. Screen recognition criteria are also used in the process of recording a macro; in this context they are sometimes called descriptors.

screen sharing
The viewing and controlling of program screens on a computer other than the user's own computer. During a screen-sharing Sametime meeting or Learning Space - Virtual Classroom Live session, one participant shares a screen with the other participants and can allow them to control the program. The program needs to be installed only on the computer of the person who is doing the actual sharing.

screen state
The set of conditions (at the time the screen was imported from the host) that determine the allowed and required processing on the screen. A screen's state operates on input to change the status, cause an action, or result in a particular output screen. A single screen can have multiple states and the allowed user actions for the screen vary depending on which state the screen is in.

screen view
In AFP Utilities, the presentation of a display shown while a user is in screen edit mode.

script
(1) A series of commands, combined in a file, that carry out a particular function when the file is run. Scripts are interpreted as they are run.
(2) The logical flow of actions for a 3270 server program.
(3) An exact text for the telesales service representative to read to a customer regarding transactions. Scripts can be short-hand or prompts to remind a representative to say certain things to a customer at certain points during a call.

scripted OS image
An unattended install action where operating system installation files that are used with some

configuration files would install the operating system on the target system using boot server technology.

scripting
A style of programming that reuses existing components as a base for building applications.

scripting language
A high-level programming, or command, language that is interpreted (translated on the fly) rather than compiled ahead of time. See also JavaScript.

script language
A high-level, application-specific scripting language that consists of statements used to develop 3270 scripts. These scripts are part of the interface between a state table and a 3270-based host business application.

scriptlet
A mechanism for adding scripting language fragments to a source file.

scroll
To move a display image vertically or horizontally to view data that is not otherwise visible in a display screen or window.

scrollability
A property of a cursor that indicates whether the cursor can fetch in a backward direction. See also fetch orientation.

scrollable cursor
A cursor that can be used to fetch in backward and forward directions. See also nonscrollable cursor.

scrollable result set
A result set that is associated with a scrollable cursor that allows the application to fetch rows and to refetch previously fetched rows.

scroll bar
A part of a window that shows a user that more information is available in a particular direction and can be moved into view by using a pointing device or the page keys.

scrolling window
The portion of the presentation space that is mapped to the viewport at any given time. The window can be moved vertically within the presentation space by scrolling. See also presentation space.

scrubbing
The removal from VOB and view storage directories of files that are no longer needed.

SCS
See SNA character string.

SCSA
See Signal Computing System Architecture.

SCSI
See Small Computer System Interface.

US006678705B1

(12) **United States Patent**
Berchtold et al.

(10) **Patent No.: US 6,678,705 B1**
(45) **Date of Patent: Jan. 13, 2004**

(54) **SYSTEM FOR ARCHIVING ELECTRONIC DOCUMENTS USING MESSAGING GROUPWARE**

(75) Inventors: **Stefan Berchtold**, Augsburg (DE); **Alexandros Biliris**, Chatham, NJ (US); **Euthimios Panagos**, Madison, NJ (US)

(73) Assignee: **AT&T Corp.**, New York, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/441,284**

(22) Filed: **Nov. 16, 1999**

Related U.S. Application Data

(60) Provisional application No. 60/108,589, filed on Nov. 16, 1998.

(51) **Int. Cl.**[7] **G06F 17/30**
(52) **U.S. Cl.** **707/204**
(58) **Field of Search** 707/204

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,890,163 A * 3/1999 Todd 707/200
5,958,005 A * 9/1999 Thorne et al. 709/202
6,199,073 B1 * 3/2001 Peairs et al. 707/204
6,263,121 B1 * 7/2001 Melen et al. 382/305

FOREIGN PATENT DOCUMENTS

JP 07046271 * 2/1995

OTHER PUBLICATIONS

Microsoft Press Computer Dictionary Third Edition, 1997, Microsoft Press, 3rd edition, pp 13, 34, 35, 196, 436, and 437.*

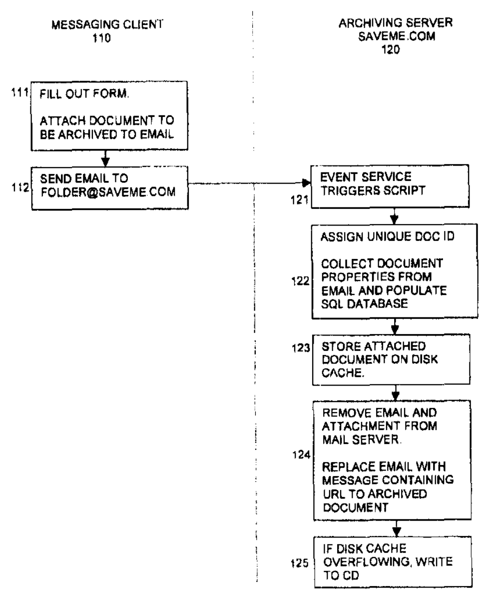
* cited by examiner

Primary Examiner—Jack Choules

(57) **ABSTRACT**

The present invention discloses an architecture for document archival built on network-centric groupware such as Internet standards-based messaging. Archiving and retrieving and classifying documents into meaningful collections is accomplished in a manner similar to sending email to recipients, retrieving messages from folders, and classifying messages into folder hierarchies. In the simplest scenario, if saveme.com is the archiving server's name, then sending an email to abc@saveme.com will cause the contents of the email message to be archived in the abc mailbox. The archived documents may be automatically stored in jukeboxes of non-tamperable media such as Write Once Read Multiple (WORM) Compact Disks (CD), which provide high storage capacity, low cost compared to magnetic disks, random data access, and long-term stability. The present invention leverages existing messaging infrastructures, and the resulting environment is not intrusive, easier to administer, and easier to deploy than conventional dedicated document archival systems.

14 Claims, 6 Drawing Sheets



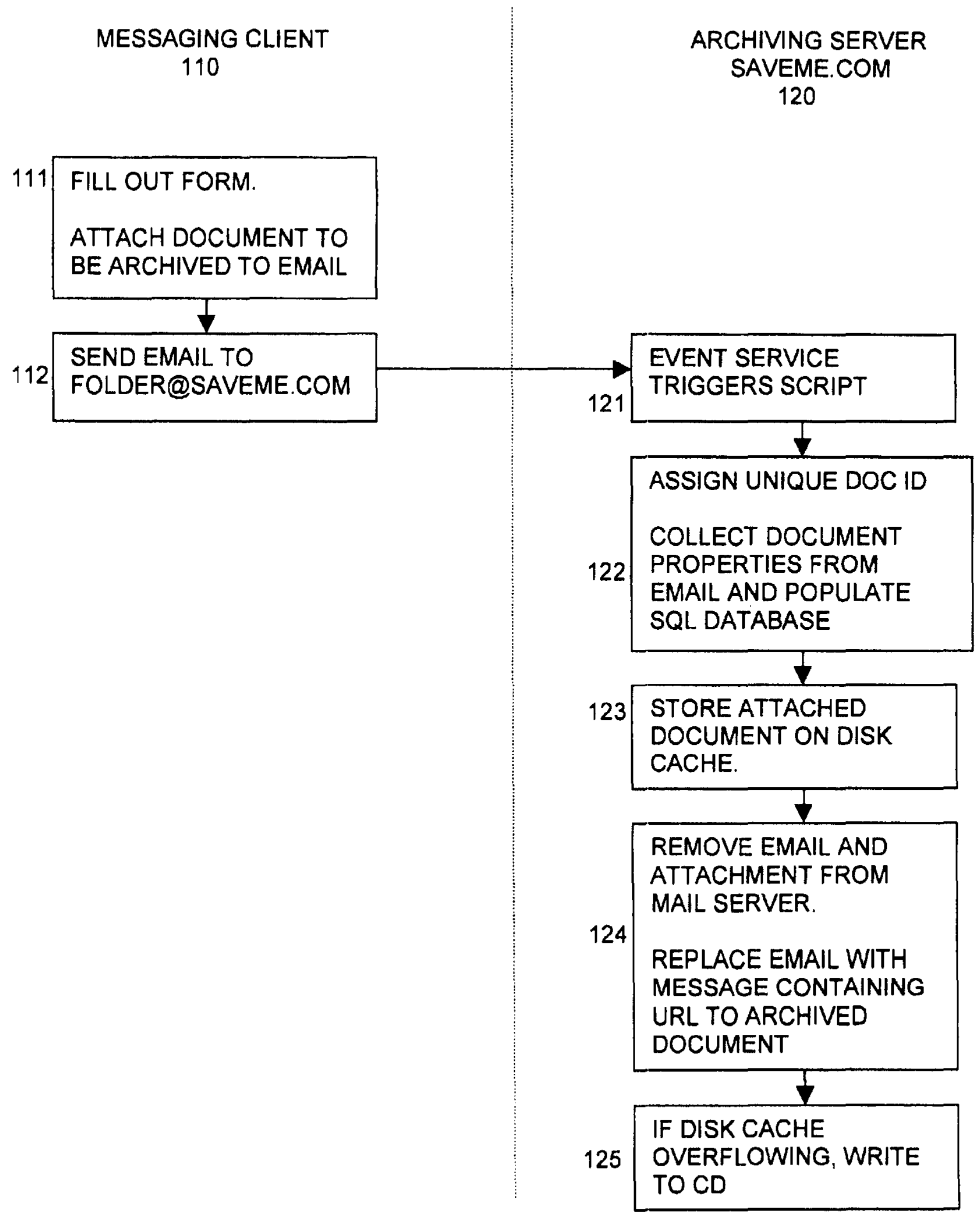

MESSAGING CLIENT
110
ARCHIVING SERVER
SAVEME.COM
120
111
FILL OUT FORM.
ATTACH DOCUMENT TO BE ARCHIVED TO EMAIL
112
SEND EMAIL TO FOLDER@SAVEME.COM
121
EVENT SERVICE TRIGGERS SCRIPT
122
ASSIGN UNIQUE DOC ID
COLLECT DOCUMENT PROPERTIES FROM EMAIL AND POPULATE SQL DATABASE
123
STORE ATTACHED DOCUMENT ON DISK CACHE.
124
REMOVE EMAIL AND ATTACHMENT FROM MAIL SERVER.
REPLACE EMAIL WITH MESSAGE CONTAINING URL TO ARCHIVED DOCUMENT
125
IF DISK CACHE OVERFLOWING, WRITE TO CD

Figure 1

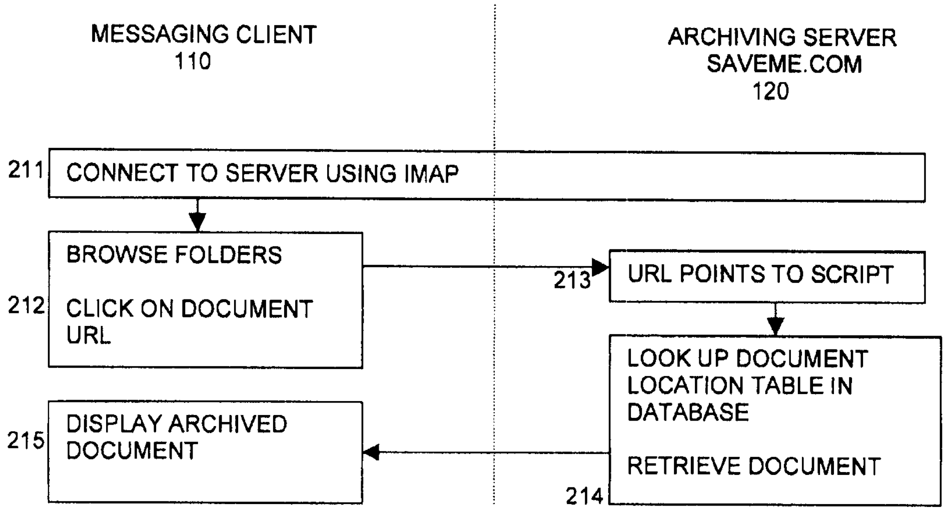


Figure 2A

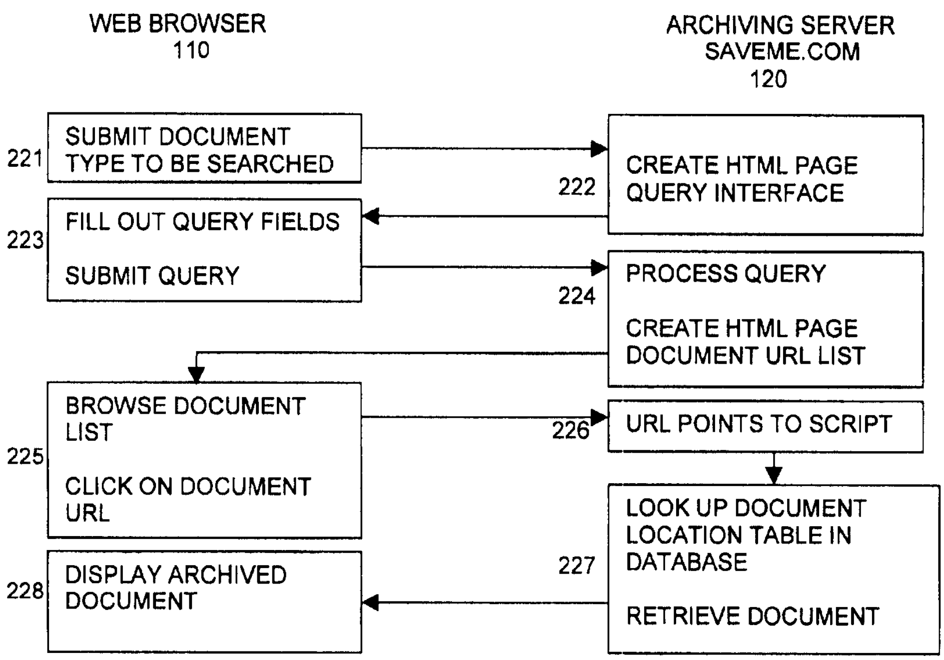
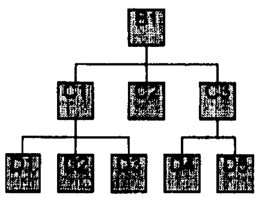
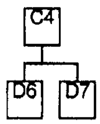


Figure 2B



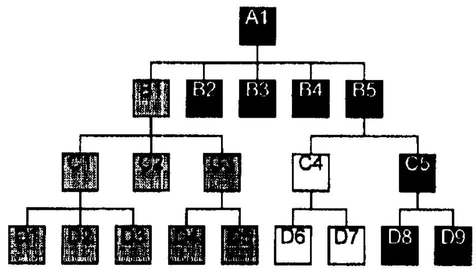
a

Figure 3A



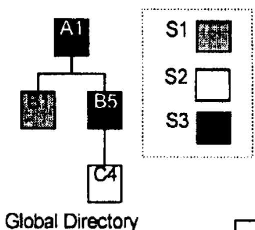
b

Figure 3B



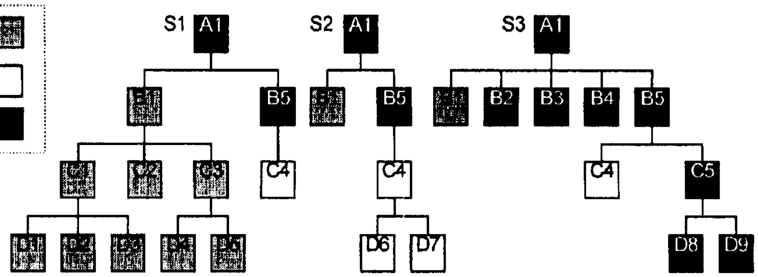
c

Figure 3C



a

Figure 4A



b

Figure 4B

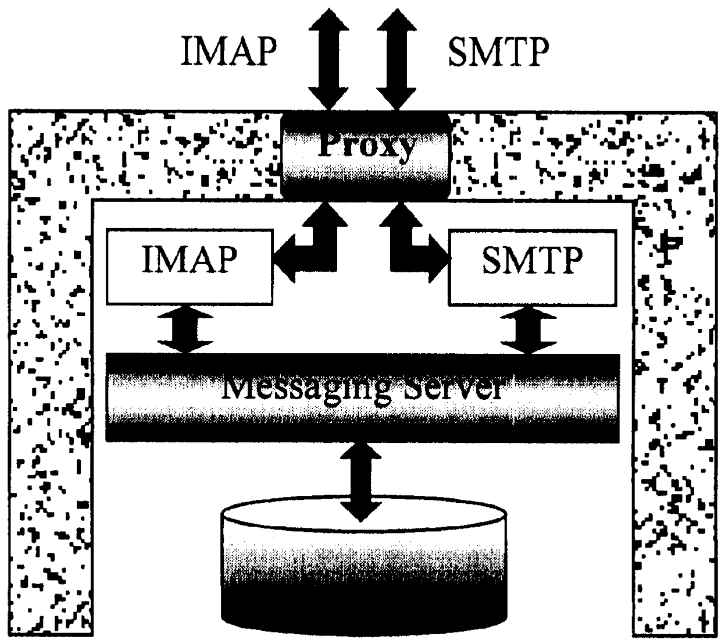


Figure 5

Voucher				
ID	no	amount	Who	what
123121	2	1,000	Sb	Pc
...	...	...	...	...
...	...	...	..	...

Purchase_order				
ID	Key	Net	Gross	HRid
124241	4291aI	2,240	2,400	1232
...	...	...	...	...
...	...	...	..	...

Shared_attributes				
ID	Classified	Email	Arrived	Dep.
46121	a/b/c	B@ab.cd	1.1.98	6177
...		...	...	...
...		...	...	...

Document_location				
ID	CD/Cache	CD no	Offset	name
46121	CD	123	127	abc
34713	Cache	n/a	n/a	a/b/c
...	...	...	...	...

Index

Figure 6

```
ShortestSubset (documents D, values V, int N)
{
    current_size = 0; start = 0; end = 0;
    while ( current_size <= 650 )           // Find initial candidate subset
        current_size += size(D_end);
        end++;
    shortest_start = start;
    shortest_end  = end - 1;

    while ( end < N )                       // Search for shorter subsets
        current_size -= size(D_start);
        start++;
        while ( current_size <= 650 )
            current_size += size(D_end);
            end++;
        if ( (V_end-1 - V_start)  < (V_shortest_end - V_shortest_start) )
            shortest_start = start;
            shortest_end  = end - 1;
    return (shortest_start, shortest_end);
}
```

Figure 7

SYSTEM FOR ARCHIVING ELECTRONIC DOCUMENTS USING MESSAGING GROUPWARE

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority to U.S. Provisional Application Serial No. 60/108,589, filed on Nov. 16, 1998, the content of which is incorporated by reference herein.

FIELD OF THE INVENTION

The present invention relates to archiving electronic documents and, more particularly, to the use of messaging systems for the archiving of electronic documents.

BACKGROUND OF THE INVENTION

Electronic archiving of documents is an increasingly important task for most large enterprises. The ease at which electronic documents can be created by modem personal computer applications has changed the way organizations store and manage critical information. In particular, most of the organizations today keep information about their core business in an ever-increasing volume of electronic documents—including product designs and documentation, corporate policies, expense reports, purchase orders, presentations, electronic messages, and data related to electronic commerce, etc. Most of these documents need to be stored for extended periods of time (i.e., archived) because they are both related to the core business and required to ensure institutional accountability. Consequently, the ability to manage, share, and control archived documents among collaborative workgroups, workflows, business partners, and across global networks is crucial in gaining advantage in today's competitive business environment.

Although there are numerous archiving software packages available in the marketplace currently, see e.g. Aegis StarView (http://www.aegisstar.com/), DocuLive (http://www.sni.com/), Lotus Domino.Doc (http://www.lotus.com/products/dominodoc.nsf), Documentum (http://www.documentum.com/), all these systems use specialized desktop applications, proprietary technology, and/or are mostly concerned with the lifetime of documents. They are difficult to administer because new software has to be installed on practically every user's desktop, possibly thousands of computers in a large organization. Also, using a document management system for the pure purpose of archiving is disadvantageous in terms of both money and complexity of the system. Individual users and IT personnel have to learn a new technology, resulting in a substantial cost for an organization in terms of training and administration.

SUMMARY OF THE INVENTION

The present invention emanates from the recognition that organizations already have substantial investments in electronic messaging technologies, which provide the most effective methods for communication, collaboration, and coordination among workers within decentralized organizations and across different companies. Messaging has evolved from a simple communications tool, which allows users to communicate with each other, to a powerful business communication infrastructure that supports collaborative computing, business process automation, electronic commerce, and distributed work environments. Messaging is also used as the middleware for accessing and controlling

shared resources, such as a printer server that receives documents as electronic mail (email) and prints a document at the appropriate printer based on the available printers, the document type, and the sender's identity.

The present invention discloses an architecture for a document archival system built on network-centric groupware such as Internet standards-based messaging. In accordance with an embodiment of the invention, archiving and retrieving and classifying documents into meaningful collections is accomplished in a manner similar to sending email to recipients, retrieving messages from folders, and classifying messages into folder hierarchies. In the simplest scenario, if saveme.com is the archiving server's name, then sending an email to abc@saveme.com will cause the contents of the email message to be archived in the abc mailbox. In another embodiment of the invention, archived documents are automatically stored in jukeboxes of non-tamperable media such as Write Once Read Multiple (WORM) Compact Disks (CD), which provide high storage capacity, low cost compared to magnetic disks, random data access, and long-term stability.

The present invention leverages existing messaging infrastructures and, thus, does not require installation of new software on every desktop. In addition, individual users and information technology personnel do not have to learn a new technology, resulting in substantial savings for an organization in terms of training and administration. The resulting environment is not intrusive, easier to administer, and easier to deploy than conventional dedicated document archival systems. These and other advantages of the invention will be apparent to those of ordinary skill in the art by reference to the following detailed description and the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagram of an archival process in accordance with an embodiment of the present invention.

FIGS. **2**A and **2**B are diagrams of archived document retrieval processes in accordance with embodiments of the present invention.

FIGS. **3**A, **3**B, and **3**C are a diagrams representing (a) a hierarchy of folders in a server embodying the present invention, (b) another hierarchy of folders in a second server embodying the present invention, and (c) a conceptual merging of the two hierarchies with a third hierarchy to form a global hierarchy.

FIGS. **4**A and **4**B are diagrams representing alternative implementations of name resolution according to an embodiment of the present invention.

FIG. **5** is a diagram of an embodiment of the present invention utilizing a SMTP and IMAP proxy architecture alternative.

FIG. **6** is an example of the relational database design in accordance with an embodiment of the present invention.

FIG. **7** is an example of an algorithm used to determine which documents to write to a CD from a disk cache.

DETAILED DESCRIPTION

In accordance with a preferred embodiment of the present invention, the archiving system is built on top of a messaging server such as Microsoft Exchange Version 5.5. Exchange supports many of the Internet protocols used pervasively today, such as the Internet Message Access Protocol (IMAP), the Simple Message Transfer Protocol (SMTP), and the Lightweight Directory Access Protocol

(LDAP). Exchange supports addressable public folders (also referred to as shared folders) on which a rich set of access permissions can be defined. In addition, the product supports server-side scripting, which allows running programs (written in JavaScript, VBScript, C++, etc.) on the server as a reaction to events occurring in public folders, such as timer events and actions like posting, editing, or deleting a message.

With reference to FIG. **1**, which sets forth an example of the archiving process in accordance with a preferred embodiment of the invention, a user utilizes a messaging client **110** to designate a document to be archived. According to standard messaging protocols, email is sent to mailboxes. A mailbox is a character string (otherwise known as an email address) which identifies a user to whom mail is to be sent. Normally, mailbox names consist of user and host specifications; the standard mailbox naming convention is defined to be user@domain. At step **112**, the user sends the document to be archived as an attachment to an email addressed to the public folder that should contain the document, e.g. abc@saveme.com, where abc is the name of a public folder on the archiving server saveme.com **120**. This can be done by specifying the folder name in any of the To:, CC:, or BCC: fields of the email. Users can use the body of the email at step **111** to specify values for some pre-determined document attributes. For example, an expense report must specify the employee submitting the report, a dollar amount, a manager to authorize the expense, etc. This can be accomplished with the use of electronic forms—pages containing the fields that must be filled out for each document type before the document submission. One such page can be designated for each document type. These forms can be stored as regular email messages in the archiving server. In order to archive a document of a particular type, the user replies to the appropriate message, fills out the form, and attaches the document she wants to archive to the message. Then, the user can send the message to the desired public folder.

All folders in the archiving server are being monitored by the event service. At step **121**, the deposit of a message in a folder triggers the Folder: : OnMessageCreated event. As a result of the triggered event, a server-side script associated with this event is invoked and is passed the folder identification and the message identifier of the newly posted message. The script at step **122** first collects the properties of the message (date, sender, etc) that can be automatically derived and assigns a unique id to the document. Then, it parses the body of the message and, if the body of the message is a form, it extracts all information contained in the form. Finally, it populates the appropriate database tables for this document. In addition, the script can also check the completeness and correctness of the submitted form by evaluating the information contained in the form and verifying that all the required ones are present (if any are missing or some constraints are violated, it can return the message to the sender and removes the message from the mail server).

Next, at step **124** the script removes the body and the attached document of the message from the email server. The document itself is stored on a separate disk cache at step **123**, waiting to be written out to a durable media storage device such as a CD. The body of the message is replaced by another message containing some information about the document and an address (e.g. a Uniform Resource Locator or URL) pointing to a script that is capable of loading the actual document. Information about the exact physical location of the document (on the storage device) is advantageously maintained in a document_location table in a SQL database maintained by the archiving server. Thus, a user can click on the URL and retrieve the document at any point in time regardless of the physical location of the document. As a next step at **125**, the script checks if the cache is overflowing; if this is the case, it invokes an algorithm to write a subset of the cache to a CD (further described below).

FIGS. **2A** and **2B** sets forth two ways by which a user can retrieve an archived document or set of archived documents, in accordance with a preferred embodiment of the invention. In FIG. **2A**, the user can navigate through the document hierarchy and manually select individual documents. Since the archive folders are merely mailboxes on an email server, a standard protocol such as IMAP can be used to access and manipulate the message folders. IMAP includes operations for creating, renaming and deleting mailboxes, checking for new messages, permanently removing messages, setting various flags associated with messages, selectively fetching portions of messages and searching existing messages. See "Internet Message Access Protocol—Version 4", RFC 1730, Network Working Group 1994. The user only has to connect to the archiving server and browse through the public folders with an IMAP compliant mail client, such as the ones offered by Netscape and Microsoft. After having found the right documents, the user can retrieve a document by clicking on the URL viewed in the message body. When the URL is de-referenced, a CGI script looks up the document-location table to determine the physical location of the message and the message is then returned to the user.

Alternatively, as set forth in FIG. **2B**, the user can declaratively specify a set of documents. The archiving system can then automatically retrieve the desired documents and deliver the documents to either a folder in the mail server or directly to a Web browser when a Web interface is used. The search process can be divided into two steps: First, the right document type to be retrieved can be chosen (with the choice of designating a search of all document types to allow for document type-independent queries). Second, the archiving system creates an HTML page that offers an appropriate query interface for this particular type of document. The user fills out the fields for the attributes that have been defined for this document type. An option for full text search can be offered. When the user clicks on the submit button, the appropriate CGI script is invoked. The entered values are checked for correct types and other constraints. Then, the query is processed (as further described below) and an HTML page containing the URLs of the actual documents meeting the query conditions is created. In order to transfer the documents to the client computer, the user merely has to click on the URLs.

There are four major administrative tasks that have to be performed in order to maintain the archiving server: adding and maintaining document types, maintaining the document hierarchy, managing access rights, and database administration. In order to add a new document type, the system administrator has to store the form for this type with the email server. She then has to add the appropriate tables in the database and implement appropriate scripts to parse the form and to make a sanity check on the information delivered by the user. The maintenance of the document hierarchy and management of access rights can be done through the tools that Exchange provides to manage the email server. The main task with respect to database administration is to add and delete appropriate indexes to the database.

Further discussed below in more detail are (a) the hierarchical naming scheme utilized in the present invention; (b)

alternative architectures to deal with shortcomings in standard Internet protocols; (c) database design issues to effectuate efficient declarative retrieval of documents; and (d) solutions to physical storage issues.

A. Naming Scheme

Given the vast amount of stored documents in an archiving system, document classification based on hierarchical organized collections is important for two reasons: First, it provides an easy and intuitive way to classify documents into collections and sub-collections—this helps focus the search into smaller domains. Secondly, it facilitates easy navigation through the collection of documents, especially when the criteria used to classify the documents in the first place (i.e., the schema of the hierarchy) are not known. There are many ways to organize documents into hierarchies. An example of such a hierarchy might be the organizational hierarchy of the corporation: business units, centers, and departments followed by the various document types. Another might be a first classification based on the year the document was archived, then the document types, followed by business units, centers, and departments. Note that a hierarchy may change over time—think of departments being renamed or split (new levels may have to be introduced in the hierarchy) and centers being merged (levels in the hierarchy may have to be merged). Also, hierarchies need not necessarily be stored (materialized). They could be computed views over the set of documents.

In accordance with a preferred embodiment of the present invention, a hierarchical naming scheme is utilized for both identifying collections of archived documents and for providing access paths to them. Documents are archived by storing them into folder hierarchies belonging to email recipients. For example, a document sent to the email address A/B/C@saveme.com, where saveme.com is the archiving server's name, will be archived in folder C, which is under folder B, in the hierarchy rooted at A. The hierarchy structure is maintained in a directory. Aliases can also be used to eliminate the need of specifying entire path names.

In a distributed system, multiple archiving servers can manage the folder hierarchy. Servers can be added, removed and reconfigured to accommodate load conditions and business considerations without breaking existing applications and user habits. This is achieved by allowing the naming components of a folder hierarchy to be resolved by different servers, supporting local autonomy. In particular, a name can be logically split into two parts, a prefix and a remainder. A directory service maintains information that maps the prefix to an archiving server responsible for managing the remainder. This process is applied recursively until the remainder is unambiguously resolved to a folder hierarchy managed by an archiving server. It should be noted that the two name parts may not consist of a fixed number of components, neither do they have to stay the same over time. Given a name, its prefix is identified by the directory service at the time the lookup is performed. A subsequent resolution for the same name may return a different prefix if the systems have been reconfigured. (The resolution scheme described is similar to the one originally used in file systems and domain name services where names are resolved incrementally from left to right, with the leftmost name component being resolved first, providing a context for the resolution of the second name, and so on.)

For example, FIG. 3A shows a document hierarchy managed by an archiving server on behalf of an organization. This hierarchy is rooted at B1 and the levels below the root could represent sub-units of B1 such as centers, departments, employees or projects that have been undertaken in B1 at some point in time. FIG. 3B shows the corresponding hierarchy for another organization rooted at C4 and managed by a second archiving server. Assuming that at some point in time an organization higher up than both B1 and C4 decides to employ the archiving service for its own needs and for some additional sub-units. The resulting configuration is shown in FIG. 3C. The first two servers still operate in an autonomous fashion for the hierarchies they manage. However, each one of them is able to resolve names that correspond to the entire hierarchy. The paths A1, A1/B1 and A1/B5/C4 are the possible prefixes that will be used for name resolution. Given a name, the longest prefix from the three above that matches a portion of the left part of the name is the prefix of that name. For example, the prefix of A1/B5/C5 is A1 while the prefix of A1/B5/C4/D7 is A1/B5/C4.

The global name resolution can be implemented as follows. Each archiving server has a directory for managing its hierarchy. A global directory keeps track the mapping between prefixes and archiving servers. Every time servers are added to or removed from the system, the global directory is updated to reflect the new mappings. However, changes to the internal structure of each directory need not propagate to the global directory as long as the path of their root to the root of the global hierarchy remains the same. FIG. 4A shows the global directory for the example shown in FIG. 3C. Every node in the hierarchy may denote a prefix (i.e., the server that maintains the hierarchy rooted at that node). However, some nodes, such as B5 in FIG. 4A, are simply intermediate paths to prefixes.

An alternative implementation would require that each server keep track of the prefixes for all other servers. FIG. 4B shows the directories S1, S2, and S3 where each one includes replicated information for the roots of the other two directories. This scheme avoids a lookup to a, possibly remote, global directory service in order to find the prefix of a name, but modifications need to propagate to each directory. As with the global directory scheme, changes to the internal structure of each directory need not propagate to the other directories as long as the path of their root to the root of the global hierarchy remains the same. The replicated scheme is better than the one that utilizes a global directory because the expected number of archiving servers that participate in a global archiving system is small, in the range of tens or even hundreds of servers for big organizations. In addition, the updates that need to propagate from one directory to the rest are also expected to be small.

B. Messaging Architecture

Although providing numerous facilities for sending, organizing and retrieving messages, Internet protocols such as SMTP and IMAP have shortcomings when it comes to supporting two features that are important for electronic archiving: (a) addressable folders anywhere in a folder hierarchy and (b) enforcing access restrictions on individual folders. In particular, IMAP does not provide sufficient access control mechanisms (although access control list extensions have been proposed), and most existing SMTP implementations do not support folder addressability. In this section, four design alternatives are disclosed based on the functionality provided by the messaging system to overcome these restrictions. The first three alternatives assume that the messaging system does not support folder addressability and access restrictions.

1. The Adjunct Approach. Using this approach, users interact directly with the messaging system—and the archival system manages the hierarchical relationships between folders and enforces access controls explicitly. The inventors

refer to this approach as being "decoupled." Since the messaging system does not support folder addressability, the following alternatives can be used for archiving documents into hierarchical folders: (a) Every folder in the hierarchy corresponds to a (virtual) user (for example, a user mailbox names a/b/c could be defined to store the folder a/b/c, another mailbox to store a/b/d, etc.); (b) All documents are sent to a single predefined mailbox, and the user specifies the actual folder path in the subject or in a user-defined header attribute (e.g., X-folder). In both cases, the archival system monitors the appropriate mailboxes, extracts the folder path from the message/folder name and forwards the message, if needed, to the server responsible for managing this path. Regarding IMAP-based navigation, email clients navigate the folder hierarchies that are dynamically constructed in response to user queries. Here, users send queries to the archival system (using a Web browser) and the archival system creates folder hierarchies for those users to store the documents that satisfy the specified selection criteria and any access restrictions that may apply.

Regarding the folders that are created, there are several policies that may be used, e.g., provide each user with a mailbox and store the answer set in the mailbox. A second alternative to navigation is to maintain role-based folder hierarchies. Each such hierarchy contains sub-folders and documents that can be accessed by the corresponding role. When a document is archived, the hierarchies correspond to roles that can access the document are updated to contain a pointer to the document. While the role-based hierarchy method works with existing IMAP implementations, it does not scale well when the number of roles increases above some threshold. In this case, the number of folder hierarchies that need to be maintained may be prohibitively large, even when certain optimizations are used. In addition, role-based hierarchies are server-specific and they are not shared across multiple archive servers. If someone needs to access documents in a remote server, an IMAP connection should be established with that server—IMAP does not support multiple servers in its current specification.

2. The Proxy-with-Mail-Server Approach. With this approach which the inventors characterize as "loosely-coupled", a proxy is placed between the users and the messaging server, as shown in FIG. 5. During document archiving, an email message containing the document is sent to an address denoting a folder hierarchy. During the recipient negotiation part of SMTP, the proxy looks up the hierarchy specified in the address of the message in a directory to determine its validity and locate the archive server responsible for this hierarchy. The directory provides DNS-like functionality, i.e.; it is be able to determine the longest prefix of an address that is handled by the server. An LDAP directory could be used for this purpose, assuming that prefix search is either implemented by the LDAP server or by a specialized wrapper, which issues multiple queries to the LDAP server. If the server is not the local one, an SMTP session can be established with the appropriate remote proxy. Otherwise, the proxy executes the following steps. First, it communicates with the archival system to get an appropriately-formatted URL to use as the message body, which would be stored with the messaging server. Next, it accepts the body of the message and stores it in the cache used for the CD jukebox. Finally, it establishes an IMAP session with the IMAP component of the messaging server and stores the modified message in the appropriate folder hierarchy.

For accessing the archived document, email clients establish an IMAP session with the local proxy. The proxy is responsible for enforcing any role-based access control restrictions that may apply to both documents and folders. In addition, the proxy may access folder hierarchies stored with remote servers, providing a unified view of the entire document archival space. The proxy relays the IMAP commands it receives from an email client to the IMAP component of the messaging system. Email clients are assumed to log onto the system using a role-name, password scheme. In addition, it is assumed that the LDAP entries that correspond to the various roles have as their mailbox entries the root folder of the hierarchy managed by the local archiving server. This implies that when a client is successfully authenticated, it can examine any folder under the root folder. The IMAP proxy enforces access control when the messaging server's IMAP server returns folder names and messages as responses to commands issued by the email client. The IMAP proxy removes from the list of returned documents the documents that are not accessible by the given role.

3. The Proxy-with-no-Mail-Server Approach. In this approach which the inventors refer to as "fully integrated", the messaging system is bypassed. The archival system communicates directly with email clients and their commands are processed internally by the archival system.

4. The Shared-Folders Approach. The last alternative, which is employed in the implementation described above, is based on the assumption that the underlying message server supports folder addressability and enforcement of access rights through public (aka shared) folders. Several messaging and groupware products, including Microsoft Exchange, Eudora WorldMail and HP OpenMail support public folders. Usually, the implementation of public folders offers a full-fledged access control mechanism on a per folder basis. The present invention, therefore, can utilize the public folder mechanism of the underlying messaging system for implementing its naming scheme and enforcing access restrictions. It should be noted, however, that shared folders do not correspond to any Internet standard and, consequently, using this approach may limit the messaging component to a small list of existing systems that offer this functionality.

C. Database Architecture and Document Retrieval

A typical archiving system used by an enterprise has to deal with huge amounts of data, practically necessitating the use of database technology. A relational database system can be used to manage the attributes associated with the archived documents, including a pointer to their physical location. As described below under section D, the actual documents are advantageously organized in a file system on CDs independently from the database system. The reason for this is that the design of commercial database systems is tightly bound to the properties of magnetic disks and, thus, CDs or even WORMs are not supported appropriately. In addition, the system should have control over the transfer of documents to CDs and their clustering.

Database Design. Since the number of archived documents is assumed to be very large, indexes require a significant space and maintenance overhead. Accordingly, the database schema must be chosen with care. As described above, a document type is a collection of attributes that describe some information about a group of documents. Obviously, given this definition, there is strong relationship between a database schema and the document types. However, there are some technical challenges: First, attributes can be either mandatory or optional. Therefore, some of the attribute values may be unknown to the system. Second, for the system, the semantics of a particular

attribute is unknown. Furthermore, for practical reasons, the unique naming of attributes cannot be strictly enforced. Thus, an attribute "x" might appear in document type "1" and in document type "2" having a different meaning. Third, the number of document types might be in the order of 1,000 whereas the total number of attributes might be in the order of 10,000 for a large system.

To avoid problems with the semantics of a query specified by the user, two mechanisms are provided: First, a user interface is provided, as described in detail above, that restricts the user to meaningful queries. Second, if a query is received that contains attributes "x" and "y", documents will be returned that are only of a type that includes attributes "x" and "y". Third, when an attribute value "x" appears in a query and "x" is not present in a given document, although "x" is in the schema of this document type, the document is not included in the result. This can occur when "x" is an optional attribute.

In the present system, a document is uniquely identified within the whole system by a document ID that is attached to each document as soon as the document arrives in the system. This is the primary key of the document. For simplification, only a relatively small number of document types (e.g. up to 1000) is allowed, which all are predefined by the system administrator. She is responsible for maintaining the set of document types and for assuring that all information relevant for searching is present in the system. She also is responsible for keeping the number of document types reasonably small.

Because of the very large number of attributes assumed to exist in the whole system, it is not feasible to use a single table for all document types. This would lead to a very sparse table since most of the attributes are unknown for most document types. Therefore, it is advantageous to assign each document type to a single table that includes the document type specific attributes. Additionally, a global table is kept that contains the attributes shared by all documents. Each table includes the document ID as a primary key. Indexes may be placed on any attribute according to the query mix and the size of the table. If a new document type is introduced into the system, the system administrator must create a corresponding table in the relational database. As the number of document types is proportional to the number of tables, this number should be reasonably small. FIG. 6 shows an example for the instant database design.

Query Processing. The following are typical queries in the archiving context that the system should be enabled to handle:

Q1. "Give me all documents that have been archived by 'Alexandros Biliris', 'Thimios Panagos', or 'Stefan Berchtold' some time in 'April 1998'".

Q2. "Give me all documents that contain the phrase 'Indexing in SaveMe'".

Q3. "Give me all documents which have been classified 'ATT/Research/Databases'".

Q4. "Give me all documents that have been archived in 'April 1998' and classified 'ATT/Research' of which the project number is 'SaveMe1' that contain the phrase 'ha, ha, hi'".

As described in detail above, queries are generally issued via an interactive query interface. The query interface performs some sanity checks such as formats of numbers or if a string represents a correct date. The query is transferred to the system as a set of attributes together with query ranges. Additionally, the query processor gets the information to what part of the hierarchy the query is restricted, and search-strings that must be contained in the resulting documents. In order to process a query that arrives in the query processor, it is necessary to consider all the three parts of the query specification: selection of attributes, selection based on the classification in the hierarchy, and full-text search. Thus, the query is split into the three parts and handled separately. Finally, the results are merged.

The full-text search portion of the query is performed first. It is assumed that a full-text search engine, such as Verity described in http://www.verity.com/products/whtpaper.htm, is available. The search engine produces a set of document IDs that point to documents satisfying the query condition, with respect to the full-text search. The selection based on the hierarchy is more sophisticated: As described in the next section, each node of the hierarchy is mapped to a string such that all nodes in a sub-tree share a common prefix. Thus, the query specification on a hierarchy can be transformed in a range query on that string. Therefore, the string can be simply included as an additional attribute "Classified" in the table of the shared attributes. For query processing, it is then possible to treat queries based on the hierarchy as range queries (prefix queries on strings can be seen as range queries) on the attribute "Classified".

As a next step, a determination is made of the set of attributes that are involved in the query. If the query specification includes the hierarchy, the attribute "Classified" is added to this set. Then, all attributes that are contained in the "Shared_attributes" table from the set are removed. Finally, a determination is made of all tables that include all attributes remaining in the set in order to generate a single SQL statement querying all affected tables. As a result of such query, a set of document Ids is again obtained. In the last step, this set of document IDs is merged with the set resulting from the full-text search, and the final result of the query is delivered back to the user. This merge step is equivalent to the intersection of two sets of document IDs. This can be done efficiently by sorting or hashing. The user has the ability to either look-up all the actual documents or to look-up only single documents. To locate a document given a document ID, the information stored in the table document_location can be utilized.

Note that an appropriate generation of indexes in the relational database system is required as in any database application. The creation of the optimal set of indexes for the given query mix and database population is the task of the system administrator as mentioned above.

Efficient Searching in Hierarchies. Several ways to index objects that are organized in a hierarchy have been discussed in the prior art. Most of these proposals focus on a small and static hierarchy where a user asks hierarchically restricted attribute queries. The most common technique is to map the position of an object in the hierarchy into a number such that all nodes in a sub-tree are assigned a number from a known disjoint interval. Then, a search on a hierarchy is reduced to a range query on this number. However, none of these proposals works on a dynamic hierarchy. Thus, the inventors propose the following simple mapping to strings instead of numbers: Suppose that each node in the hierarchy is assigned a label such that sibling nodes have different labels. If we want to denote the position of a node N in the hierarchy, we simply concatenate the labels on the path from the root node to N adding a "/" to each label. Thus, we get a string of the form "a/b/ . . . /x". If one wants to select all documents in a certain sub-tree (rooted in a node S having a label "s"), one simply has to query the database for all documents that have assigned a position "a/b/ . . . /s/*". This translates to a range ["a/b/ . . . /s", "a/b/ . . . /t"). If a

document arrives in the system, the user will classify it. This means, it is assigned to a node in the hierarchy. If the document's information is inserted in the database, the according string defining the position of the document in the hierarchy is computed as described above and this attribute ("Classified") inserted with the other attributes into the according tables. In order to process a query based on the hierarchy, the sub-tree to which the query is restricted must first be determined. Then, a determination is made of the string defining the position of the root node of this sub-tree in the hierarchy, and a query is made of the database with a range query as described above.

Note that the strings that identify a position in the hierarchy may become very long. This however is not a particular problem because modern relational database systems can handle large strings easily and even the implemented index technology (usually prefix B+-trees) is not severely affected. As an extension of the above-described technique one might want to use multidimensional index technology in order to answer combined queries more efficiently. This however invokes the problem of having a multidimensional index structure storing both string and numerical attributes.

D. Physical Storage

A preferred embodiment of the present invention uses a durable media such as optical disks or CDs rather than magnetic disks as the underlying physical storage device. Apart from the legal reasons for storing data on durable media, magnetic disks are not an economical option for storing a large number of documents over a long period of time. A good physical organization of the data on the optical storage system is essential for an efficient document retrieval (search time and throughput), CD storage utilization, and bulk-drop of expired documents. Usually, optical devices contain between 100 and 1000 media (e.g. CDs), each storing about 650 Mbytes, keeping a total amount between 50 Gbytes and a few TerraBytes. A typical CD Jukebox has four reading and one writing devices and a single robot arm loading and unloading the devices. In contrast to magnetic disks, the time to access a new CD, which is not currently mounted in one of the drives, might be very long (in the order of a minute). Thus, each single retrieval process of documents should be restricted to as few CDs as possible.

The above-described implementation had the following storage hardware available: main memory in the order of 1 Gbyte, secondary storage on magnetic disks in the order of 10 Gbytes and optical storage (CD Worm) in the order of 100 Gbytes. The secondary storage is used to store the database including all indexes and as a cache for the optical disks. As CDs must be written in a single pass, it is necessary to cache 650 Mbytes on magnetic disc for each CD to be written. Thus, the available cache restricts the number of CDs that can be written simultaneously. (Given the above amounts of storage, this number is in the order of 15 disks.) Note that, although most CD writers technically allow for more than one write-process on a single CD, each additional write costs a large space overhead. Furthermore, CDs written in one step tend to be more stable over time. Therefore, it is advisable not to make use of this feature.

In order to define a good strategy to assign documents to CDs, one has to consider the following points:

1. Documents expire after an arbitrary period of time such as 1 year, 5 years, 10 years, or never. Sometimes expiration must force document deletion. For example, financial organizations need to hold credit card data for seven years, but legally they are not allowed to retain it after that time.

2. It cannot be assumed that archiving of all documents expiring in a certain year can be done in a short period of time. Rather, documents expiring in 2007 will probably be archived during 1997, 1998 and 1999.

3. In order to avoid the system to be filled with expired documents and to maintain a high storage utilization, a mechanism should be provided that allows one to discard entire CDs if all the documents stored on this CD are expired. The inventors refer to this deletion of a large set of expired documents as a "bulk-drop".

If one looks at all these technical restrictions, there are two different criteria to optimize: On the one hand, documents should be clustered on CDs according to their expiration date; on the other hand, documents should be clustered that are frequently retrieved simultaneously. Furthermore, the query mix is unknown in advance and will definitely change over the years such that it is unknown a priori which documents will be requested together. Therefore, the inventors propose the following solution: The cache is primarily organized as a set of documents. Each document D_i in the cache is assigned a value v_i that can be used to cluster the document to disk. This value typically is the expiration date of the document. However, it could also be a value such as project number or even a combination of both. The documents in the cache are sorted according to these values v_i. A heap can be used to maintain the sorted set. If the cache overflows, a subset of the documents in the cache of size 650 Mbytes should be selected and written to disk. The algorithm set forth in FIG. **7** selects the shortest subset (with respect to the values v_i) containing no more than 650 Mbytes by only making a single pass through the documents stored in the cache. For clarity of the presentation, FIG. **7** omits some necessary checks for array boundaries. The algorithm is a greedy algorithm and does not necessarily lead to a globally optimal solution. However, given the various technical restrictions, an optimal solution is far from being feasible. Thus, the algorithm gives a sufficiently good solution. Furthermore, at any point in time, it is possible to reorganize a subset of documents by collecting the appropriate documents from CDs and writing new CDs. For this process of reorganizing n CDs, only a single 650 Mbyte cache is required.

The foregoing Detailed Description is to be understood as being in every respect illustrative and exemplary, but not restrictive, and the scope of the invention disclosed herein is not to be determined from the Detailed Description, but rather from the claims as interpreted according to the full breadth permitted by the patent laws. It is to be understood that the embodiments shown and described herein are only illustrative of the principles of the present invention and that various modifications may be implemented by those skilled in the art without departing from the scope and spirit of the invention.

What is claimed is:

1. A computer readable medium containing executable program instructions for performing a method of archiving electronic documents on a computer comprising the steps of:

receiving a document via a messaging protocol, the document being an attachment to an electronic mail;

determining to which mailbox on a server the document is addressed; and

automatically archiving the document if the mailbox is designated as an archive folder, the archiving comprising the steps of:

storing the document on a storage device;

deleting the document and the electronic mail from the mailbox; and

storing a message in the mailbox comprising an address enabling access to the stored document.

2. The computer readable medium of claim 1 wherein the messaging protocol is the Simple Message Transfer Protocol.

3. The computer readable medium of claim 1 wherein the message in the mailbox can be accessed by the Internet Message Access Protocol.

4. The computer readable medium of claim 1 wherein the address is a URL to a script which retrieves the document from the storage device.

5. The computer readable medium of claim 1 further comprising the step of storing document attributes and a location of the document on the storage device in a database.

6. The computer readable medium of claim 1 wherein the storage device is an optical media jukebox with a magnetic disk cache.

7. An electronic document archival system comprising:

a hierarchy of addressable folders; and

a system for receiving electronic mail via a messaging protocol further comprising:

means for archiving documents addressed to a folder in the hierarchy, the archiving means further comprising means for extracting a folder path in the hierarchy from an electronic mail; and

means for enforcing access control restrictions that apply to the folders or the contents of the folders.

8. The electronic document archival system of claim 7 wherein the system is a messaging system and the folders are public folders in the messaging system.

9. The electronic document archival system of claim 7 wherein the system is a proxy server in communication with a messaging system via a messaging protocol.

10. The electronic document archival system of claim 7 wherein the system is fully integrated with a messaging system.

11. A method of archiving documents on a durable medium comprising the steps of:

storing the documents in a cache;

assigning to each document a clustering value;

sorting the documents according to the clustering values;

when the cache overflows, selecting based on the clustering values a subset of the documents with a total size no larger than that of the durable medium and storing the subset of the documents on the durable medium.

12. The method of claim 11 wherein the clustering value comprises a document expiration date.

13. The method of claim 11 wherein the subset is shortest with respect to the clustering values.

14. The method of claim 11 wherein the durable medium is an optical compact disk.

* * * * *

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re *Inter Partes* Reexamination of:	§	Control Number: 95/001,398
	§	
U.S. Patent No. 6,684,789	§	Art Unit: 3992
	§	
Attorney Docket No: 38512.24	§	Examiner: Mary J. Steelman

For: METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER

DECLARATION OF PAUL JACOBS UNDER 37 C.F.R. § 1.132

Mail Stop *Inter Partes* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

I, Paul Jacobs, declare as follows:

1. I have been retained by the Requester for the above referenced reexamination proceedings of U.S. Patent No. 6,684,789 ("the '789 patent"). I have reviewed the '789 patent. I have also reviewed the Office action issued in the reexamination of the '789 patent on November 19, 2010 ("OA"), as well as the Patent Owner's Amendment And Response ("PO Remarks") and the supporting declaration of David Birnbaum, Ph.D. ("Birnbaum Dec.") filed on March 30, 2011.

2. I am familiar with the level of ordinary skill in the art with respect to the alleged invention of the '789 patent. The analysis below are my findings based on the level of ordinary skill in the art at a time period at and prior to May 11, 2001. My qualifications and findings are set forth below and the resume attached at the end of this declaration. I have been compensated in this matter at my customary hourly rate.

I. QUALIFICATIONS

3. I received a Bachelor of Science in Applied Mathematics from Harvard University in 1981, a Masters of Applied Mathematics from Harvard University in 1981, and a Ph.D. in Computer Science from the University of California at Berkeley in 1985.

4. I am the founder and president of Jake Technologies, Inc., a software consulting practice where I specialize in advanced information management systems, including search engines and automated text processing. Prior to becoming a consultant in 2002, I was employed for 17 years in the computer industry as a researcher and software executive.

5. I have authored or co-authored over 50 scientific and technical publications, I am listed as an inventor on two U.S. patents directed to computational lexicons, and I have over 25 years of experience in the computer and information industry.

6. I have served in numerous professional and scientific capacities, including one year as a visiting professor of computer science at the University of Pennsylvania and several years as a member of the executive committee of the Association for Computational Linguistics. Currently, I serve on the Public Policy Council of the Association for Computing Machinery (USACM). I am currently an adjunct lecturer at the University of Maryland in College Park, where I have taught classes in the College of Information Studies (The "iSchool") since 2007.

II. FINDINGS

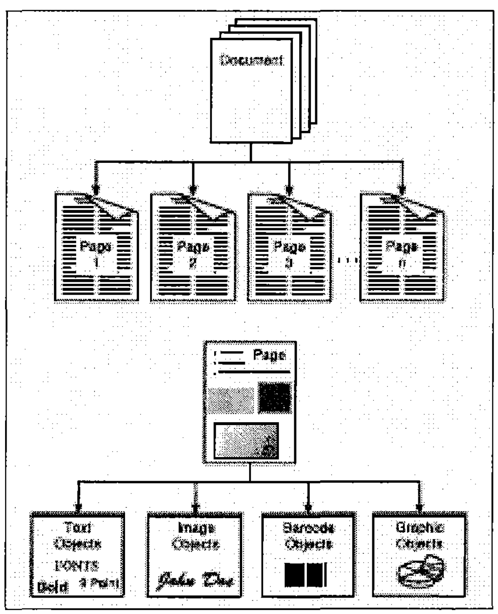
7. The findings below are in light of the ordinary skill in the art at the time period at and prior to May 11, 2001.

8. I have read the IBM reference, and am familiar with the teachings of this reference. I also am familiar with the AS/400, the system upon which the IBM reference is based.

9. The Advanced Function Presentation /Printing (AFP) architecture described in the IBM reference describes receiving many different types of print data streams, including streams referred to as "SCS" and "IPDS" as well as "PostScript." (IBM at 14-16.) IBM clearly teaches that one or more of these streams is "optionally parsed," as that term is commonly used in the art. (IBM at 194.) The ordinary meaning of "parsing" is not limited to syntactic parsing using a formal grammar or similar restriction. I understand that the PO argues that parsing in the '789 Patent claims means "a syntax analyzer: it is capable of analyzing the syntax of an input data stream based on a formal grammar," and that SCS and IPDS streams do not require that type of parsing. (PO Remarks at 6.) Even under the PO's construction, however, there is no reason why IPDS and SCS print data streams are not parsed, as recited in the claim.

10. Birnbaum's analysis that IBM's SCS and IPDS data streams do not have a syntax is not correct. (Birnbaum Dec. ¶ 12-23.) The existence of structured fields, and the ability to parse using a state machine are not determinative of whether a data stream has syntax. In fact, parsers are frequently used to separate commands with structured fields and state machines have been used for parsing even complex languages such as English. Birnbaum also draws his conclusion because processing print lines "can be done by a series of if-then statements." (Birnbaum Dec. ¶ 20.) In fact, a series of if-then statements could be sufficient for even the most complex parsing tasks.

11. The PO construes "object-oriented" in the '789 Patent claims as requiring "that objects have the ability to be organized, for example, in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend." (PO Remarks at 10.) The IBM reference is an object-oriented architecture under that construction. The IBM reference teaches hierarchy of objects, such as is shown in Figure 129, reproduced below. A "document" object is at the top, "page" objects are in the middle of the hierarchy, and image, text, barcode, and graphics objects are embedded within the page. Therefore, this clearly shows that IBM teaches that a document is composed of "hierarchy" of objects—"object-oriented" even within the PO's construction of the claim.



IBM, Fig. 129

In addition, I refer to a 1997 document titled "Mixed Object Document Content Architecture", which describes the data stream as used by AFP, as referenced in the IBM reference. Selected documents are included in Exhibit V, and show that the IBM objects support inheritance.

> *Hierarchical Defaults*: Parameter values established by an environment group at a higher level in the document component hierarchy will be the default for a subordinate level unless a value is specified at the subordinate level. ... The placement of parameter values at a higher level in the document hierarchy, for the purpose of enabling lower levels to <u>*inherit*</u> these values as defaults, is known as *factoring*.

(Ex. V, at 10; underlining added.[1])

12. I am also very familiar with the use of scripts. The PO and Dr. Birnbaum assert that scripts must be non-compiled structures. It is further noted that the '789 patent refers to different kinds of scripts, including scripts in the programming language "Javascript." Scripts written in many languages, including Lisp, Javascript, Perl, and Python are regularly compiled using either ahead-of-time (AOT) or just-in-time (JIT) compilation. These script compilers were all in common usage at or before May 11, 2001. For example, see Exhibit R, "Server-Side JavaScript Guide:"

> If the HTML and JavaScript files contain server-side JavaScript, you then compile them into a single JavaScript application executable file. The executable is called a web file and has the extension .web. The JavaScript application compiler turns the source code HTML into platform-independent bytecodes, parsing and compiling server-side JavaScript statements. (JavaScript Guide at p. 39)

13. It is a well-known principle that compiled scripts often run faster than non-compiled scripts, but non-compiled scripts can run on more systems. People of ordinary skill in the art routinely choose whether to use compiled or non-compiled scripts, dependent upon their specific situation. I see no reason why any aspect of a script described in the '789 patent claims requires non-compiled scripts, instead of compiled scripts.

---

[1] This document was obtained from an archival website over the Internet. The website is provided at the beginning of the document, and shows a 1997 copyright date. Only selected portions of this document have been provided.

14. The IBM reference also teaches that DDS commands are assigned to specific objects. Specifically, IBM refers to "record level" and "field level" DDS commands. Record level DDS commands are assigned to the entire document or print stream. Field level DDS commands are assigned to individual objects. (See IBM at 127-128; Fig. 63.) One example of a DDS script taught by IBM is the text-rotate script: "TXTRTT." (IBM at 134.) This script is assigned to one of the text objects (*see* text objects shown in Fig. 5 of IBM, reproduced above) so that the text prints at a different orientation. The result of assigning this DDS command to the text object "ROTATE" is shown in IBM Fig. 63, reproduced below. This command is clearly assigned to a particular object, because not all of the text or other objects in Fig. 63 are rotated.

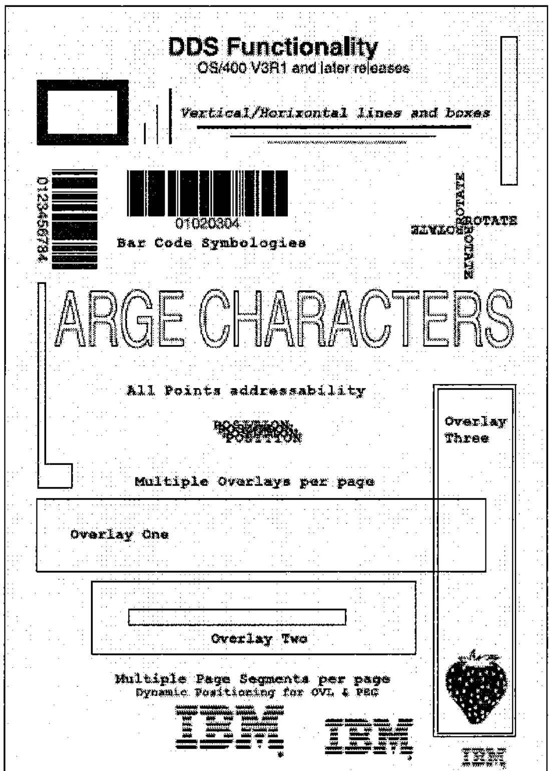


Figure 63. DDS Functionality

Text-rotate "TXTRTT" is assigned to this text object.

Text-rotate is not assigned to this text object

IBM's use of DDS commands is consistent with the well-known and common practice of using scripts to enhance the functionality of objects.

15. I have also read the Interleaf reference, the Interleaf patent, and the Lieberman patent. Each of these references refers to scripts, and the common usage of scripts in various

software scenarios. Scripting is a general-purpose, or multi-purpose software technique, and is not limited or restricted to one area of software technology. A person of ordinary skill in the art would be able to know and use scripts in many different situations. Consistent with this fact, the scripts described in the '789 patent, IBM, Interleaf, the Interleaf patent, and Lieberman all operate in a predictable manner. Using a type of script in one of these references in any other reference would function in its known, predictable manner. For example, the "Server-Side JavaScript Guide" shows examples of scripts also being used to manipulate files (Ex. R at pp. 170-177), send email messages (Ex. R at pp. 167-170), access databases (Ex. R at pp. 189-262), dynamically load functions and libraries (Ex. R at pp. 178-181), and other various functions. Exhibit S, U.S. Pat. No. 6,678,705, has examples of scripts (in various languages, including Javascript) sending and receiving email (Ex. S at 3:8-4:30), retrieving information from Internet servers (Ex. S at 6:49-8:41), as well as interacting with various databases (Ex. S at 8:50-11:23).

16. Interleaf describes transforming a "document file stream" into active documents. (Interleaf at 84.) The PO has made several arguments that a document file stream is not the same as a "print data stream," as that term is used in the '789 patent. I have been asked to consider this scenario, and whether it would be obvious for a person of ordinary skill in the art at the time of the filing of the '789 patent to use Interleaf technology with a print data stream (as shown in IBM) instead of a document file stream. The answer is yes, for several reasons.

17. First, Interleaf and IBM both describe a computer system receiving a data stream, and performing the exact same operations as done in the '789 patent. ('789 patent at 8:40-48.) The received data stream performs the same function in each reference. Both references also refer to performing various operations on the transformed data stream, as discussed in the OA.

18. Second, IBM lists "PostScript" as an example input print data stream. (IBM at 16.) It would have been obvious to use a PostScript data stream instead of a Lisp-oriented document stream—as Interleaf teaches—because the syntax of PostScript was based on and inspired by Lisp. (*See e.g.*, Ex. W at 1, "The first few chapters of this book help you put PostScript into perspective by comparing it to languages you probably already know, such as C. There are stronger similarities between PostScript and, say, Forth or Lisp, but if you are expert in either of those languages you probably won't have much trouble mastering PostScript.")

19. Third, PostScript is useful for both displaying and printing documents. (Ex. U, "PostScript [is] used for displaying and printing," *see also*, Exhibits W and X.) In fact, I am aware of computer systems in the 1980s that received print data streams for displaying and editing documents. For example, the Sun Microsystem "NeWS" computer and the "NeXT" computer both used a version of PostScript display systems in the mid-1980's. (*See, e.g.*, Ex. X, wikipedia.org/wiki/Adobe_PostScript.) As noted above, the '789 patent lists PostScript as one example of an input print data stream.

20. Thus, a person of ordinary skill in the art would have reason to use a print data stream in the Interleaf system.

21. I declare that all statements made herein on my own knowledge are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 or Title 18 of the United States Code.

4/25/2011
Date

Paul Jacobs

2. On July 16, 2010, a request for *inter partes* reexamination of the '789 patent was filed by a third party requester, and it was assigned Reexamination Control No. 95/001,398 (the '1398 proceeding).

3. On October 14, 2010, the Office issued an order granting *inter partes* reexamination in the '1398 proceeding, indicating that an Office action would issue in due course.

4. On November 19, 2010, the Office issued a first Office action on the merits.

5. On January 10, 2011 patent owner timely filed a response to the first Office action on the merits.

6. On February 9, 2011, third party requester submitted comments after patent owner's January 10, 2011 response and the November 19, 2010 first Office action. Concurrently, third party requester filed the present petition paper in the '1398 proceeding entitled, "PETITION UNDER 37 C.F.R. § 1.183 TO WAIVE PAGE LIMIT REQUIREMENT FOR THIRD PARTY REQUESTER COMMENTS UNDER 37 C.F.R. § 1.943(b)."

7. On March 15, 2011, the Office issued a notice of defective paper indicating that patent owner's January 10, 2011 response was improper and setting a time period for resubmission.

8. On March 30, 2011, patent owner submitted a corrected response to the first Office action, addressing the issues raised by the Office in the March 15, 2011 notice of defective paper. Concurrently, patent owner filed a petition requesting waiver of the regulatory page limit requirement of 37 CFR 1.943(b).

9. On April 13, 2011, the Office dismissed as moot requester's February 9, 2011 petition.

10. On April 26, 2011, third party requester submitted comments after the patent owner's March 30, 2011 response and the November 19, 2010 first Office action. Concurrently, third party requester filed a new petition paper in the '1398 proceeding entitled, "PETITION UNDER 37 C.F.R. § 1.183 TO WAIVE PAGE LIMIT REQUIREMENT FOR THIRD PARTY REQUESTER COMMENTS UNDER 37 C.F.R. § 1.943(b)."

DECISION

I. Relevant Statutes, Regulations and Procedures

37 CFR 1.183 provides:

> In an extraordinary situation, when justice requires, any requirement of the regulations in this part which is not a requirement of the statutes may be suspended or waived by the Director or the Director's designee, *sua sponte*, or on petition of the interested party, subject to such other requirements as may be

ACTION CLOSING PROSECUTION (37 CFR 1.949)	Control No. 95/001,398	Patent Under Reexamination 6684789
	Examiner MARY STEELMAN	Art Unit 3992

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --

Responsive to the communication(s) filed by:
Patent Owner on 30 March, 2011
Third Party(ies) on 06 September, 2011

Patent owner may once file a submission under 37 CFR 1.951(a) within 1 month(s) from the mailing date of this Office action. Where a submission is filed, third party requester may file responsive comments under 37 CFR 1.951(b) within 30-days (not extendable- 35 U.S.C. § 314(b)(2)) from the date of service of the initial submission on the requester. **Appeal cannot be taken from this action.** Appeal can only be taken from a Right of Appeal Notice under 37 CFR 1.953.

All correspondence relating to this inter partes reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of this Office action.

PART I. THE FOLLOWING ATTACHMENT(S) ARE PART OF THIS ACTION:

1. ☒ Notice of References Cited by Examiner, PTO-892
2. ☒ Information Disclosure Citation, PTO/SB/08
3. ☒ IDS 04/26/2011

PART II. SUMMARY OF ACTION:

1a. ☒ Claims 1-82 are subject to reexamination.
1b. ☐ Claims _____ are not subject to reexamination.
2. ☐ Claims _____ have been canceled.
3. ☐ Claims _____ are confirmed. [Unamended patent claims]
4. ☐ Claims _____ are patentable. [Amended or new claims]
5. ☒ Claims 1-82 are rejected.
6. ☐ Claims _____ are objected to.
7. ☐ The drawings filed on _____ ☐ are acceptable ☐ are not acceptable.
8 ☐ The drawing correction request filed on _____ is: ☐ approved. ☐ disapproved.
9 ☐ Acknowledgment is made of the claim for priority under 35 U.S.C. 119 (a)-(d). The certified copy has: ☐ been received. ☐ not been received. ☐ been filed in Application/Control No _____
10. ☐ Other _____

U.S. Patent and Trademark Office
PTOL-2065 (08/06)

Paper No. 20101018

INTER PARTES REEXAMINATION OF USPN 6,684,789 B2 to Krautter.

ACTION CLOSING PROSECUTION (ACP)

This office action addresses reexamination of claims 1-53 of USPN 6,684,789 B2 to Krautter. Per Non Final Office Action (11/19/2010), claims 1-53 are rejected. Per Patent Owner's request (03/30/2011, p. 1), claims 54-82 are added. Original claim 9 is amended. Patent Owner states, "The claims do not enlarge the scope of the patent or introduce new matter. Support for the new claims is provided at Appendix A." Claims 1-82 will be reexamined.

Patent Owner (p. 1) has noted the following litigation: CCP systems AG v. Samsung Electronics Corp., Ltd. Samsung Electronics America, Inc., Samsung Networks, Inc. and IBM Corp., 2:09-cv-04254 (D. N.J.). Patent Owner states, "In the litigation, Samsung has admitted its printers have included "JScribe Core software or software adapted from JScribe Core software." (Complaint ¶17, attached as Appendix B- received 03/30/3011). Patent Owner asserts the '789 patent explains, JScribe®, CCP's copyrighted and trademarked software, is covered by the patent in this reexamination (see '789 patent, Col. 5, lines 55-59 and Picht Dec. ¶11-20)

This Action Closing Prosecution Office Action fully responds to Patent Owner Response (03/30/3011), the Declaration of Roland Widuch (01/10/2011), Chief Executive Officer of CCP, the Declaration of Christoph Picht (03/30/2011), Director of Business Development at CPP, and

the amended Declaration of David Birnbaum, Ph.D. (03/30/2011). The declarations are entered into the record and addressed separately below. This Action Closing Prosecution Office Action fully responds to Third Party Requester Comments (09/06/2011) and the 37 CFR 1.132 Declaration of Paul Jacobs (09/06/2011). For clarity, it is noted that throughout Patent Owner's Remarks, Patent Owner has referred to Third Party Requester as "Samsung."

Newly added Prior Art

To propose rejections to Patent Owner's newly added claims, Third Party Requester has added new art. (See Requester Comments 09/06/2011, pp. 32-36 and IDS received 04/26/2011.)

USPN 6,678,705 B1 to Berchtold et al.; file date Nov. 16, 1999; issue date Jan. 13, 2004.

Declarations entered into the record

The **Declaration of Roland Widuch** (01/10/2011, incorporated by reference), has been considered and entered into the prosecution record. The Widuch Declaration is directed to assertions of commercial success and licensing and appears to be a declaration under 37 CFR 1.131. The Widuch Declaration additionally cites to the following evidence:

Exhibit A – JScribe® Print Server Solution 1.0 IBM, build 1.02, undated

Exhibit B – Alliance in Printing, Samsung and IBM Korea, undated

Exhibit C – dated 01/26/2006, non-English document, @ p. 3, "BestSeller Award 2005: Kategorie Service: CCP Systems AG, ONVENTIS GmbH"

Exhibit D - Chinict 2007 Top ICT Innovators Press Release "CCP Systems wins the European Commission ChinICT Innovator Award"

Exhibit E – "New Samsung MFP with advanced features offers easy integration for efficient, simple and reliable printing, January 2009

Exhibit F – JScribe Samsung White Paper, undated

Exhibit G – "Our thoughts on designing printers and printing solutions", dated 12/15/2000 URL<http://www.samsung.com/us/b2b/learning/whitePapers.html> (Single page screen shot.)

Mr. Widuch discloses an extensive background in the software printing business, does not disclose monetary compensation related to the declaration, and as the CEO of CPP Systems AG ("CCP"), is considered to have a vested interest in the outcome of this reexamination.

The **Declaration of Christoph Picht** (03/30/2011, incorporated by reference), in support of the Patent Owner, has been considered and entered into the prosecution record. Mr. Picht, a Director of Business Development at CCP Systems AG ("CCP), a longtime employee, with an extensive background in software development presents credentials of one skilled in the art. Mr. Picht does not disclose compensation received and is otherwise silent, but presumed to have a vested interest in the outcome of the reexamination. Mr. Picht asserts (¶7) that CCP and IBM entered into a license in July 2004 that gave IBM a "license under any patents ... to make, have made, use [etc.] ...that included the '789 patent. Mr. Picht asserts (¶¶9-10) that JScibe® is a software product that CCP had licensed to IBM Corporation's German subsidiary. IBM's Korean subsidiary, in turn, sublicensed the JScribe® software to Samsung Electronics Corp., Ltd., Korea. Mr. Picht opines (¶11) "Claim 1 of the '789 patent covers the JScribe® software when used for printing. JScribe® reads in an input print stream that is analyzed by a parser. This stream is split up into graphically representable objects which are then stored in a memory in an object-oriented format. JScribe® software and any of its functionalities can be activated by scripts which can be developed easily by users, either with CCP's Software Development kit (SDK), or with any other editor, and then be deployed to the JScribe® enabled device where they

become active." ¶¶12-14 note script usage with JScribe®. ¶15 asserts that claims 2-16 of the '789 patent incorporate JScribe® functionality into Samsung printers. ¶¶16-20 further assert opinions that Samsung printers use JScribe® software as covered by the patented claims of '789; Samsung has incorporated the JScribe® software or parts of it into many of its printers and multifunctional devices. Mr. Picht notes (¶21) corresponding patents in Europe and Japan (EP 1282883 B1 and JP 3974782 B2). As noted in the Petition Decision (08/23/2011, p. 5), the Picht declaration presents only objective and technical opinion evidence and is free of legal argument.

The **Declaration of David Birnbaum, Ph.D.** (03/30/2011, incorporated by reference), in support of Patent Owner has been considered and entered into the prosecution record. The Birnbaum Declaration has presented credentials (¶¶1-4 and CV, Exhibit A, received 03/30/2011) demonstrating one of skill in the art. It is presumed that the declaration is filed under 37 CFR 1.132. It is noted that Dr. Birnbaum does disclose (¶7) compensation received and is otherwise silent regarding any vested interest in the outcome of the reexamination. ¶¶8-11 provide a brief summary of the prior art. ¶¶12-41 provide an opinion on the definition and functionality of the claimed "parser," "object oriented format," "scripts," "script assigned to an object." ¶¶42-44 argue the rejection of claim 20 and opine that an AS/400 is not a printer and it would not be obvious to import the functionality of the AS/400 into a printer. Dr. Birnbaum addresses (¶¶45-52 & 97-114) the anticipation rejections of dependent claims as rejected under prior art IBM and under Interleaf. Dr. Birnbaum addresses obvious rejections (¶¶53-79, 80-96, 115-116, 117-129, 130, and 131-133) under combinations of IBM, Interleaf, Interleaf Patent, and Lieberman. The

Birnbaum Declaration introduces into prosecution the following supporting Exhibits (all received 03/30/3011):

Exhibit A – David Birnbaum curriculum vitae

Exhibit B - The IBM iSeries Primer Device Programming, Version 5, Document No. SC41-5713-05, 2002 (cover page, copyright page, TOC iii-viii and pp. 492-98)

Exhibit C "Printer Device Programming" V4, Release 3 Fourth Edition, May 1999 at 556. (TOC pp. iii-ix, pp. 61 & 555-558)

Exhibit D – IBM Terminology, 09 December 2010; URL<http://www-01.ibm.com/software/globalization/terminology/s.html> 3 pages

The 37 CFR 1.132 **Declaration of Paul Jacobs** (received 09/06/2011, Ex T, incorporated by reference), legal representative of Third Party Requester, has been considered and entered into the prosecution record. The Jacobs declaration has presented credentials (¶¶3-6) demonstrating one of skill in the art. A resume as noted in ¶2 is not found in the papers submitted. It is noted that Paul Jacobs does disclose (¶2) compensation received and is otherwise silent regarding any vested interest in the outcome of the reexamination. The Jacobs Declaration cites to the following evidence (all entered 09/06/2011), as noted in Appendix A:

Exhibit R - Server-Side JavaScript Guide, version 4.0. Sun Microsystems, 1999.
Exhibit S - USPN 6,678,705 B1 to Berchtold; filed Nov. 16, 1999; issued Jan. 13, 2004
Exhibit U - "Computer Dictionary," Microsoft Press, Third Edition, 1997, defining "PostScript."
Exhibit V - "Mixed Object Document Content Architecture," Ref. Doc. No. SC31-6802-04, File Number S370-40, Copyright IBM Corp. 1990, 1997, (selected portions)
Exhibit W - Reid, "Thinking In PostScript," 1990 (selected pages)
Exhibit X - en.wikipedia.org/wiki/AdobePostScript; a review of the term "PostScript."

<u>Submitted Non Patent Literature and Other Evidence</u>

Patent Owner has included Appendix A – D in Remarks (03/30/2011) as follows:

Appendix A – Support for new claims 54-82, pp. 1-20.

Appendix B - Case 2:09-,v-04354-DMC, pp. 1-68 CCP SYSTEMS AG, Plaintiff, v. SAMSUNG ELECTRONICS CORP., LTD., SAMSUNG ELECTRONICS AMERICA, INC., SAMSUNG NETWORKS, INC. and IBM CORPORATION, Defendants.

Appendix C – IBM Dictionary of Computing, 1993, defining the term "object" at p. 471.

Appendix D – A comparison of patent classifications 101/484 versus 717/139.

Evidentiary submissions by Patent Owner and Third Party Requester have been considered. The degree of consideration to be given to such information will be normally limited by the degree to which the party filing the information citation has explained the content and relevance of the information. Citations to non-patent literature that are appropriate for printing on the first page of a patent are listed on the attached Form PTO-892. **Examiner** notes that for brevity, citations to law and the MPEP found in the arguments, declarations, and rebuttals have not been repeated, but are incorporated by reference.

The issue raised (Third Party Comments, 09/06/2011, bottom of p. 28) by Requester is outside the scope of reexamination, and otherwise duly noted.

Claim Interpretation

Dr. Birnbaum opines (¶¶31-38) on the term "**script**" and (¶¶39-41) the phrase "**script assigned to an object**" (...A script is a program or series of commands that is interpreted and runs in real time rather than compiled and then executed...[a] high-level programming, or command, language that is interpreted (translated on the fly) rather than compiled ahead of time...interpreted as they are run." See Exhibit D. In contrast to scripts, programs written in

standard program languages, such as FORTRAN, C++, etc., must typically be compiled into object code before being executed. Using scripts provides at least two advantages: flexibility and portability... may be freely modified, including at run-time, which is not true for compiled code. For example, in one embodiment of the invention, the data processing unit permits stored objects, including particularly script objects, to be read out graphically, to be changed, to be deleted or to be appended...cannot be done using compiled object code. Object code is limited because it is platform-dependent. A program written in a compiled language must be compiled into object code for a particular processor architecture. In contrast, any processor capable of interpreting a script can run the script. This renders scripts portable, i.e., platform-independent. This means the DDS in IBM is compiled before being used for printing, can only be executed as object code after being compiled, and, unlike scripts, must be recompiled if modified. This feature of DDS is a critical distinction from a script, in which interpretation and execution of the script are typically simultaneously triggered, in the case of the '789 patent, for example, by the incoming print data. Regarding a "script assigned to an object," [e]ven assuming that the "elements" in IBM (images, text, etc.) were objects stored in object-oriented format, and that the DDS used in IBM is a script, the DDS is not assigned to an element, but rather is applied to the document as a whole).

> **Exhibit D**-IBM Terminology: script (1) A series of commands, combined in a file, that carry out a particular function when the file is run. Scripts are interpreted as they are run. (2) The logical flow of actions for a 3270 server program. (3) An exact text for the telesales service representative to read...

Dr. Jacobs opines (¶¶12-13) on the term "**script**" (...'789 patent refers to different kinds of scripts, including scripts in the programming language "Javascript." Scripts written in many

languages, including Lisp, Javascript, Perl, and Python are regularly compiled using either ahead-of-time (AOT) or just-in-time (JIT) compilation. These script compilers were all in common usage at or before May 11, 2001...It is a well-known principle that compiled scripts often run faster than non-compiled scripts, but non-compiled scripts can run on more systems. People of ordinary skill in the art routinely choose whether to use compiled or non-compiled scripts, dependent upon their specific situation. I see no reason why any aspect of a script described in the '789 patent claims requires non-compiled scripts, instead of compiled scripts.)

Examiner broadly interprets a script as a set of instructions to an application or a utility program, that may be compiled or not compiled (i.e., interpreted). Examiner agrees that scripts are not necessarily interpreted. See Microsoft Dictionary Fifth Edition:

> script n. A program consisting of a set of instructions to an application or a utility program. The instructions usually use the rules and syntax of the application or utility. On the World Wide Web, scripts are commonly used to customize or add interactivity to Web pages. See also macro.
>
> scripting language n. A simple programming language designed to perform special or limited tasks, sometimes associated with a particular application or function. An example of a scripting language is Perl. See also Perl, script.

JScribe® - As excerpted from the '789 specification (5: 55-6: 5), "At this point, it should be mentioned that these systems operating by the method according to the invention are also designated JScribe (registered trademark) systems and, accordingly, the method according to the invention is also designated JScribe (registered trademark)." "When JScribe (registered trademark) is used, developers and system houses will therefore be in a position to provide objects and functions which are stored in resident form in the printing system and permit and control desired individual operating sequences. These objects and functions can use any

functionality provided by the JScribc (registered trademark) basic technology, including extremely demanding commands for the job or page processing and for the complete control of the print data and emulations. The method according to the invention preferably also enables access to internal printer functions and status information (page counter, network components, file system and so on), for example via a script." '789, 6: 38-46, "For example, the simple download of JScribe (registered trademark) sequences (scripts with appropriately associated objects) can, for example, arrange for the printer automatically to fetch information about current share prices, to format it and to print it out. Image information, text documents, web pages, XML documents and any other desired print data can be analyzed while dispensing with any preparation by the PC (for example by a printer driver), modified if necessary and printed out in optimum quality."

Examiner understands JScribe scripts can direct **a networked computing system** to fetch information, format/transform the information and control output to an output device. Graphic objects that have been detected by a parser reading an input data stream (input print data stream) and stored in memory can be read out via an application interface (FIG. 1, from device 3 to device 8), changed, deleted or appended (7: 14-16), combined into super-objects(4: 10-11), split up into part objects of lower complexity (4: 46), i.e., transformed. Script objects ('789, 7: 47-51) also may be "read out graphically, to be changed, to be deleted or to be appended..." The objects are transformed into a format for the control of an output device ('789, 5: 6-11 & 3: 5-20: a printer or archiving devices, folding systems, enveloping systems or security equipment...all the devices needed in the widest sense for document processing), combined into an output print data

stream and **are output to memory**. (emphasis added) One embodiment discloses a graphically representable object stored in the memory in the object-oriented format is assigned at least one script which controls external devices ('789, 5: 3-10). Other embodiments disclose that a script assigned to a stored graphically representable object may encode commands for receiving, sending, of forwarding data (image, text, web pages, XML documents, e-mail) ('789, 5: 11-31), and in some embodiments automatically by use of a timer ('789, 6: 25-37). The script language (a single standardized programming language) may be used in a heterogeneous network, in lieu of device specific driver code ('789, 5: 38-55).

JScribe® uses objects and functions which are stored in memory and controls desired individual operating sequences, including commands for the job or page processing, and for the complete control of the print data and emulations. Also, via script, access to internal printer functions and status information is enabled. ('789, 7: 1-2 & 7: 55-56: "...print system...to be programmed...")

See '789, 6: 52-65, "**The language used for the scripts according to the present invention is preferably Java Script**. Java Script, as **a world-established standard** for the script-controlled, intelligent programming of web pages, has triggered in the Internet an avalanche of innovative and functional solutions which have contributed decisively to ringing in the age of eBusiness and eCommerce. This intelligent technology, which has so decisively marked the worldwide, rapid development of the Internet, is therefore now also available for printing systems for the first time and here preferably **forms the basic technology for script applications** in the area of the present

invention, and consequently **print and document management**, which is certainly uniquely and, as compared with established solutions, considerably more cost-effective." (emphasis added)

Examiner understands the term "**output**" to be in reference to storing a transformed data stream to memory ('789, 3: 5-20, FIG. 1, #7 to #6, output to memory) **or alternately** the term "output" also appears to be used in reference to sending a transformed document to an output device within a printing system ('789, 9: 23), preferably laser printing systems and digital copying systems, sent as e-mail or else transferred to archiving systems (FIG. 1. #3, from a PC computer or "intelligent output device" to a separate device #9). **Output device** ('789, 5:6-10) is broadly described as "archiving devices, folding systems, enveloping systems or security equipment...devices needed in the widest sense for document processing."

Examiner broadly constructs the term "**Printer Server.**" The term is not defined in the '789 specification. See 8: 24-28: "The system according to the invention can also be integrated into a printer or else a printer server.... implemented on PC server platform." Examiner will interpret a "printer server" to be a computing device programmed with the methods as described in the '789 patent. Claim 21 appears to define the term "characterized in that it has a system for the transformation of digital print data streams as claimed in claim 17." Notably the term "**printer**" in some instances appears to be interchangeable with "printer server," as it is defined identically in claim 20. **Alternately** the term "**printer**" clearly refers to a printer device ('789, Abstract, "output device (9), preferably a printer).

Dr. Birnbaum opines (¶¶12-23) on an interpretation of the term "**parser**" (...is a syntax analyzer...capable of analyzing the syntax of an input data stream based on a formal grammar. ...functionality permits the parser to analyze even complex page description languages (PDLs), in comparison to a state machine driven by regular expressions, which can only reasonably handle single commands in a line by line approach.)

Dr. Jacobs (¶¶9-10) rebuts the narrow interpretation by Dr. Birnbaum asserting: "The ordinary meaning of "**parsing**" is not limited to syntactic parsing using a formal grammar or similar restriction." "...parsers are frequently used to separate commands with structured fields and state machines have been used for parsing even complex languages such as English.

Examiner cites to the '789 specification evidencing the term. In reference to the term "**parser**," see '789 2: 64-66, "...recognizing more complex page description languages, whose syntax may no longer be described with the aid of simple regular expressions..." At '789 3: 21-30, "In this case, as opposed to the use of a status machine, the analysis and splitting of the input print data stream by a **parser (syntax analyzer)** ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used. Instead, such a parser, in terms of its theoretical performance, corresponds to a Turing machine and therefore ensures the theoretically maximum achievable performance for the analysis and splitting up of formal languages." At '789, 8: 61-64, "...the input print data stream 2 is analyzed and split up by a parser 4. The graphic objects 5, 5a

recognized as the product of this splitting..." **Examiner** asserts that the term "**parser**" or "**syntax analyzer**" is not narrowly defined in the specification. **Examiner** asserts a broad understanding of the term "**parser"** as a software algorithm that recognizes input data (instructions, commands, syntax, grammar, including objects etc.) and separates meaningful portions of the data stream. See "parser" addressed in ACP Response to Argument section, beginning at p. 92 below.

Patent Owner has provided Appendix C – IBM Dictionary of Computing, defining the term "**object**" that appears in several instances to be specific to particular architectures or applications:

> (1) In computer security... (2) A passive entity that "contains or receives data; for example, bytes, blocks, clocks, fields, files, directories, displays, keyboards, network nodes, pages, printers, processors, programs, records, segments, words. Access to an object implies
> access to the information it contains. (3) in SAA Common User Access architecture... (4) In AIX Enhanced X-Windows... (5) in the AIX object data manager... (6) In programming
> languages, a data object, (7) In AIX graphics... (8) in the Network Computing System...(9) In the IBM ImagePlus...(10) In object-oriented design or programming, an abstraction consisting of data and the operations associated with that data. See also class. (11) In the AS/400 system, a named storage space that consists of a set of characteristics that describe itself and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed; for example, programs, files, libraries, and folders...(12) In SQL...(13) See arithmetic object, compound object, data object, integral object.

Examiner opines that "object" definitions at 10 and 11 may be meaningful to this office action. Examiner agrees that broadly an object in object-oriented design or programming is an abstraction consisting of data and the operations associated with that data.

Dr. Birnbaum opines (¶¶24-30) on the term "**object oriented format**" ("object-oriented programming" uses objects...data, properties and methods, together with their interactions as defined by the methods... typically include features such as class hierarchy, data abstraction, encapsulation, modularity, polymorphism, and inheritance... objects have the ability to be organized, for example, in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend.

Dr. Jacobs agrees (¶11) that "object-oriented (the ability to be organized...in hierarchical classes...objects inherit properties...)."

Examiner in general agrees with the term as presented. The concepts of object oriented programming (format) are well known in the art.

See discussion related to "**digital print data stream"** in ACP, beginning at p. 148 below.

Claim construction in litigation and the claim construction during a reexamination need not be consistent because the standards governing claim interpretation by the Courts and by the Office are different. *See* MPEP § 2686.04 II. Courts presume that a claim is valid and is given its plain and ordinary meaning. *See* 35 U.S.C. § 282 (2009); *Schumer v. Lab. Computer Sys., Inc.,* 308 F.3d 1304, 1316 (Fed.Cir.2002). During a reexamination proceeding, however, the Office makes no such presumption and applies the broadest reasonable interpretation to a claim in light of the specification. *See* MPEP §§ 2111, 2686.04 II; *In re Trans Texas Holding Corp.,* 498 F.3d 1290,

1298 (Fed. Cir. 2007) (citing *In re Yamamoto,* 740 F.2d 1569, 1571 (Fed.Cir.1984)). The rationale underlying the "broadest reasonable interpretation" standard used by the Office is that it reduces the possibility that a claim, after issuance or certificate of reexamination, will be interpreted more broadly than is justified. *See* MPEP § 2111.

Patent Owner may choose to amend the specification to correct a clear error. See '789, 9: 2-4, "In this way, the objects 5, 5a stored in the memory 6 can, for example, be displayed on a screen 9 and modified as desired." "Screen 9" should be corrected to read "screen 8," as shown in FIG. 1.

<u>Requester Proposed New Grounds of Rejection to Address New Claims</u>

Re. **Ground #1**: In the OA, claims 1-5, 17, 20-26, and 38-41 stand rejected as being anticipated by IBM under 35 U.S.C. 102(b). The Requester proposes that Ground 1 should now include new claims 54, 56, 59, 61, 62, 68, 70, 71, 76-78, and 80-82.

Re. **Ground #2**: In the OA, claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, and 48-49 stand rejected as being obvious over IBM in view of Interleaf under 35 U.S.C. 103(a). The Requester proposes that Ground 2 should now include new claims 54, 56, 59, 61, 62, 68, 70, 71, and 76-82. Therefore, claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, 48-49, 54, 56, 59, 61, 62, 68, 70, 71, and 76-82 should be rejected under 35 U.S.C. 103(a) as being obvious over IBM in view of Interleaf.

Re. **Ground #3**: In the OA, claims 1-14, 16-18, 20-35, 37-51, and 53 stand rejected as being obvious over IBM in view of Interleaf and the Interleaf Patent under 35 U.S.C. 103 (a). The

Requester proposes that Ground 3 should now include new claims 54, 56, 57, 59-62, 64-66, 68-74,

and 76-82. Therefore, claims 1-14, 16-18, 20-35, 37-51, 53-54, 56, 57, 59-62, 64-66, 68-74, and 76-82 should be rejected under 35 U.S.C. 103(a) as being obvious over IBM in view of the Interleaf Patent.

Re. **Ground #4**: No change is proposed. In the OA, claims 15, 19, 36, and 52 stand rejected as being obvious over IBM in view of Interleaf and further in view of the Interleaf Patent

and Lieberman under 35 U.S.C. 103(a).

Re. **Ground #5**: No change is proposed. In the OA, claims 1-2, 4-7, 11-12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 stand rejected as being anticipated by Interleaf under 35 U.S.C. 102(b).

Re. **Ground #6**: No change is proposed. In the OA, claims 15, 36, and 52 stand rejected as being obvious over Interleaf in view of Lieberman under 35 U.S.C. 103 (a).

Re. **Ground #7**: In the OA, claims 1-2, 4-14, 15-18, 22-23, 25-35, 41-51, and 53 stand rejected as being obvious over Interleaf in view of the Interleaf Patent under 35 U.S.C. 103(a). Therefore, claims 1-2, 4-14, 15-18, 22-23, 25-35, 41-51, 53-54, 57, 59, 61, 68, 70, and 76-82 should be rejected under 35 U.S.C. 103(a) as being obvious over Interleaf in view of the Interleaf Patent.

Re. **Ground #8**: In the OA, claims 15, 19, 36, and 52 stand rejected as being obvious over Interleaf in view of the Interleaf Patent and Lieberman under 35 U.S.C. 103(a). Therefore,

claims 15, 19, 36, 52, and 58 should be rejected under 35 U.S.C. 103(a) as being obvious over Interleaf in view of the Interleaf Patent and Lieberman.

Re. **Ground #9**: In the OA, claims 1-53 stand rejected as being obvious over IBM in view of Interleaf, the Interleaf Patent, and Lieberman under 35 U.S.C. 103(a). Therefore, claims 1-53 and 58 should be rejected under 35 U.S.C. 103(a) as being obvious over IBM in view of Interleaf, the Interleaf Patent, and Lieberman.

New Ground #10: The Requester proposes that new claim 55 should be rejected under 35 U.S.C. 103(a) as being obvious over Interleaf in view of the Interleaf Patent and the '705 patent.

New Ground #11: The Requester proposes that new claims 55, 63, 67, and 75 should be rejected under 35 U.S.C. 103(a) as being obvious over IBM in view of Interleaf, the Interleaf Patent, and the '705 patent.

New Ground #12: The Requester proposes that new claims 54, 56-62, 64-66, 68-71, 73-74, 76, 77, and 80-82 should be rejected under 35 U.S.C. 103(a) as being obvious over the Interleaf Patent in view of IBM.

New Ground #13: The Requester proposes that new claims 55, 63, 67, and 75 should be rejected under 35 U.S.C. 103(a) as being obvious over the Interleaf Patent in view of IBM and the '705 patent.

New Ground #14: Claims 1-2, 4-7, 11-12,.22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 should be rejected as being obvious over Interleaf in view of IBM under 35 U.S.C. 103(a). These claims currently stand rejected as being anticipated by Interleaf, and this ground is

presented

in the alternative, as discussed above beginning at pg. 16 & 23 of these Comments.

New Ground #15: A discussion of how the new claims fail to meet the statutory requirements of Section 112 is provided in section B) below. Claims 56 and 59-67 should be rejected for being indefinite or non-enabling under 35 U.S.C. 112.

New Ground #16: Requester understands that Section 101 rejections are not applied during reexamination; however, 37 CFR 1.906 allows the Requester to make note of this issue, and requests that the Examiner note this issue in the next office action.

Analysis of Proposed Third Party Requester's Rejections

Per Third Party Comments 09/06/2011 (pp. 21-38), an SNQ to include new claims 54-82 has been raised by the IBM reference, Interleaf, Interleaf Patent, Lieberman, and USPN 6,678,705 B1 to Berchtold et al. as follows:

Re. **Ground #1**: Requester's proposed rejection of new claims is **adopted.** The original rejection is modified to include new claims 54, 56, 59, 61, 62, 68, 70, 71, 76-78, and 80-82. **Claims 1-5, 17, 20-26, 38- 41, 54, 56, 59, 61, 62, 68, 70, 71, 76-78, and 80-82 are rejected under 35 U.S.C. 102(b) as anticipated by IBM.** Pertinent discussions in the Request found on pages 7-8 & 10-12 and Exhibit G (07/16/2010) and Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference.

Re. **Ground #2**: Requester's proposed rejection of new claims is **adopted.** The original rejection is modified to include new claims 54, 56, 59, 61, 62, 68, 70, 71, and 76-82. Claims 1-7, 11-12,

17, 20-28, 32-33, 38-44, 48-49, 54, 56, 59, 61, 62, 68, 70, 71, and 76-82 are rejected under 35 U.S.C. 103(a) as being obvious over IBM in view of Interleaf. Pertinent discussions in the Request found on pages 7-8 & 10-12 and Exhibit H (07/16/2010) and Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference.

Re. **Ground #3**: Requester's proposed rejection of new claims is **adopted.** The original rejection is modified to include new claims 54, 56, 57, 59-62, 64-66, 68-74, and 76-82. Claims 1-14, 16-18, 20-35, 37-51, 53-54, 56, 57, 59-62, 64-66, 68-74, and 76-82 are rejected under 35 U.S.C. 103(a) as being obvious over IBM in view of Interleaf and further in view of Interleaf Patent. Pertinent discussions in the Request found on pages 10-12 and Exhibit I (07/16/2010) and Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference.

Re. **Grounds #4 - #6**: No change (Requester Comments 09/06/2011, p. 22) to the rejection is proposed. (Requester at p. 28 proposes that claim 70 should be rejected under Ground #5, as anticipated by Interleaf under 35 USC 102(b). Such a rejection is improper due to dependence on claim 68, which is rejected under Grounds 1-2, 7, and 12. Examiner presumes this is a typo.)

Re. **Ground #7**: Requester's proposed rejection regarding new claims is **adopted.** The original rejection is modified to include new claims 54, 57, 59, 61, 68, 70, and 76-82. Claims 1-2, 4-14, 15-18, 22-23, 25-35, 41-51, 53-54, 57, 59, 61, 68, 70, and 76-82 are rejected under 35 U.S.C. 103(a) as being obvious over Interleaf in view of the Interleaf Patent. Pertinent discussions in the Request found on pages 13-15 and Exhibit K (07/16/2010) and Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference.

Re. **Ground #8**: Requester's proposed rejection of new claim 58 is **adopted.** The original rejection is modified to include new claim 58. Claims 15, 19, 36, 52, and 58 are rejected under 35 U.S.C. 103(a) as being obvious over Interleaf in view of the Interleaf Patent and Lieberman. Pertinent discussions in the Request found on pages 13-19 and Exhibit K (07/16/2010) and Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference.

Re. **Ground #9**: Requester's proposed rejection of new claim 58 is **adopted as modified.** See Non Final Office Action 11/19/2010, p. 40: "Claims 1-53 are rejected under 35 USC 103(a) as obvious over the Interleaf Patent in view of IBM. This rejection proposed by the Third Party requester in the request for inter partes reexamination at pages 7-8 and 16-17 and claim chart (Exhibit L) is adopted as proposed in the request. See Request p. 16-17 and Exhibit L." (Requester 09/06/2011, p. 22 incorrectly states that Ground #9 is a rejection based on obviousness over IBM in view of Interleaf, the Interleaf Patent, and Lieberman) The proposed rejection of new claim 58 is modified to be rejected under the combination of Interleaf Patent in view of IBM, the proper combination of prior art under Ground #9. Pertinent discussions in Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference. Claims listed in proposed Ground #12 are consolidated to Ground #9: Claims 54, 56-60, 62, 64-66, 68-71, 76-77, and 80-82.

Re. **Ground #10**: Requester's proposed rejection of new claim 55 is **adopted.** New prior art '705 is applied to new claim 55. Claim 55 is rejected under 35 U.S.C. 103(a) as being obvious over Interleaf in view of the Interleaf Patent and the '705 patent. Pertinent discussions in Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference.

Re. **Ground #11**: Requester's proposed rejection of new claims is **adopted.** New prior art '705 is applied to new claims 55, 63, 67, and 75. Claims 55, 63, 67, and 75 are rejected under 35 U.S.C. 103(a) as being obvious over IBM in view of Interleaf, the Interleaf Patent, and the '705 patent. Pertinent discussions in Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference.

Re. **Ground #12**: Requester's proposed rejection of new claims is **adopted as modified.** Rejections using the combination of Interleaf Patent in view of IBM have been previously adopted under the original Grounds of Rejection #9. (See Non Final Office Action 11/19/2010, p. 40: "Claims 1-53 are rejected under 35 USC 103(a) as obvious over the Interleaf Patent in view of IBM.) The proposed rejection is modified to include new claims 54, 56, 57-62, 64-66, 68-71, 74, 76, 77, and 80-82 and is consolidated under Grounds of Rejection #9. Claims 73-74 are not properly rejected under the Ground #9 combination of prior art. Claims 73-74 (dependent on claim 72, rejected under Ground #3- IBM, Interleaf, Interleaf Patent) are rejected under Ground #3. Pertinent discussions in Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference.

Re. **Ground #13**: Requester's proposed rejection of new claims is **adopted as modified.** New prior art '705 is applied to new claims 55, 63, and 67. Claims 55, 63, and 67 are rejected under 35 U.S.C. 103(a) as being obvious over Interleaf Patent in view of IBM, and the '705 patent. New claim 75, dependent on claim 72, cannot be included in this rejection as claim 72 is rejected under the combination of IBM, Interleaf, and Interleaf Patent. New claim 75 is rejected under Ground #11 (IBM in view of Interleaf, the Interleaf Patent, and the '705 patent). Pertinent

discussions in Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference.

Re. **Ground #14**: Requester's proposed new grounds of rejection are **not adopted.** New grounds of rejection presented in the alternative are not adopted because Examiner maintains that processing an input data stream of a document in preparation for being output to printing (as taught is Interleaf) is analogous to the claimed transformation of a data stream, of a document, that may be sent to a printer (claimed as input print data stream that is transformed...).

Re. **Ground #15**: Requester's proposed rejections of claims 56 and 59-67 as indefinite or non-enabling under 35 U.S.C. 112 are **adopted as modified**. Claims 56-67 and 74 are rejected under 35 USC 112, paragraph 2. See ACP, 35 USC 112 rejections below.

Re. **Ground #16**: Requester (Requester Comments 09/06/2011, pp. 23-24) has raised an issue related to 35 USC 101. Examiner notes that this issue is outside the scope of reexamination.

Claim Rejections - 35 USC § 112

The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

Claims 56-67 and 74 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 56, 62, 66, and 74 are rejected under 35 U.S.C. 112, second paragraph, as being incomplete for omitting essential steps, such omission amounting to a gap between the steps. See MPEP § 2172.01. The omitted steps are an enabling disclosure showing how to print without a printer driver. Patent Owner support (Appendix A 03/30/2011) cites to '789 5: 38-41: "Systems that operate on the method according to the invention, such as printing systems are capable...of printing original print and image data without a printer driver."

Claim 57 recites the limitation "via the application interface" in lines 3-4. There is insufficient antecedent basis for this limitation in the claim.

Claim 58 is rejected due to dependence on rejected claim 57.

Claim 59 recites the limitation "the cases defined in the script" in the last limitation. There is insufficient antecedent basis for this limitation in the claim. It is unclear what "the cases" refers to.

Claim 59 is unclear as to how a printer can output a print data stream.

Claims 60-63 are rejected due to dependence on claim 59.

Claim 64 is unclear as to how a printer can output a print data stream.

Claims 65-67 are rejected due to dependence on claim 64.

<u>Rejection Under 35 U.S.C. 314(a) - Claim Enlarges Scope of Patent</u>

Claims in an inter partes reexamination proceeding will not be permitted to enlarge the scope of the claims of the patent.

Claims 59-63, 64-67, 68-71 and 72-75 are rejected under 35 U.S.C. 314(a) as enlarging the scope of the claims of the patent being reexamined. 35 U.S.C. 314(a) states that "no proposed amended or new claim enlarging the scope of the claims of the patent shall be permitted" in an inter partes reexamination proceeding. A claim presented in a reexamination "enlarges the scope" of the patent claims where the claim is broader than the claims of the patent. A claim is broadened if it is broader in any one respect, even though it may be narrower in other respects.

Regarding independent claim 59, the broadest patented claim based on a "printer" (claim 20/17/1) recites the limitations "data processing unit" and "communications interface." These limitations are not present in newly proposed claim 59. Thus, newly proposed claim 59 is broader in scope. Dependent claims 60-63 fail to rectify the omission and are therefore rejected under the same reasoning.

Regarding independent claim 64, the claim for "a printing system" does not appear to be based on any original claim. Thus, newly proposed claim 64 lacking support in original claims enlarges the scope of the patent. Dependent claims 65-67 fail to rectify the omission and are therefore rejected under the same reasoning.

Regarding independent claim 68, the broadest patented claim based on a "printer server" (claim 20/17/1) recites limitations the "data processing unit" and "communications interface." Thus, newly proposed claim 68, lacking these limitations is broader in scope. Dependent claims 69-71 fail to rectify the omission and are therefore rejected under the same reasoning.

Regarding independent claim 72, the claim for a "printing system" does not appear to be based on any original claim. Thus, newly proposed claim 72 enlarges the scope of the patent. Dependent claims 73-75 fail to rectify the omission and are therefore rejected under the same reasoning.

Statutory Basis for Claim Rejections – 35 USC § 102 and 103

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Ground #1: Claims 1-5, 17, 20-26, 38-41, 54, 56, 59, 61, 62, 68, 70, 71, 76-78, and 80-82 are rejected under 35 USC 102(b) as anticipated by IBM. The Request at pp. 7-8 & 10-12 and

Exhibit G (07/16/2010) claim charts are incorporated by reference. See Requester Comments 09/06/2011, p. 21, regarding Ground #1, which are incorporated by reference.

Regarding **claims 1, 17, 22, and 38**, IBM describes the Advanced Function Presentation and printing system (AFP). (IBM, p. 1): a. reading "lines of output" sent from a print application and read into the AFP [reading an input print data stream]. (IBM, p. 193-194) IBM teaches (p. 14) a SNA character string (SCS) data stream and the Intelligent Printer Data Stream (IPDS) [print data stream]. See also p. 405, “spooling”, the reading of input data streams. b. the lines of output are "optionally parsed into individual fields." [analyzing the input data stream by means of a parser] (IBM, p. 194-195) c. the AFP's parser splits the lines of output into "print fields". [splitting the input data into the graphically representable objects] (IBM, p. 194) An example of a graphically representable object / print field is a bar code. (Id.) See also IBM, p. 96, 214, & 276-AFP print resources are objects, such as overlays and page segments. d. AFP is an "object-oriented system," with the objects being stored in "printer memory." [storing the graphically representable objects in a memory in an object-oriented format] (IBM, p. 7-8, 15-font objects, 22-23) See p. 21, “AFP…is based on the MO:DCA-P (Mixed Object: Document Content Architecture-Presentation) standard…the architecture defines how mixed objects (image, graphics, fonts, overlays, and bar codes) are to be pulled together and presented as a single page or document." At p. 21, “Page elements…are defined once [storing the graphically representable objects in a memory in an object-oriented format] and then referenced as Required…” AFP uses scripts referred to as "Data Description Specification," or DDS. [the object-oriented format stored in the memory including at least one stored script] (IBM, pp. 115, 127-128; stored script @ p. 40) As an example, a DDS script [stored script assigned to object

oriented format] called "ZFOLD" instructs the printer [transform into a format for the control of an output device / printer] to fold the printed paper twice, to form the shape of a "Z." (IBM, pp. 134-135) IBM teaches (pp. 8, 15-17, 135) that DDS is used to define application output by building the data fields to be printed, transforming the format of digital print data streams. Executed printer output is described at p. 193. IBM teaches (pp. 1, 403, 405) OS/400 (the operating system for the AS/400 processor) and PSF/400 (the AFP system software for AS/400 printers) which reads on the computer readable medium limitations of claim 22. The executing operating system code and the PSF/400 code read on the "computer data signal embodied in a carrier wave representing sequences of instruction…" feature of claim 38. IBM teaches (p. 404) an attached hardware device. [control of an output device]

Regarding **claims 2, 23, and 39**, IBM teaches (p. 218), "Documents are made up of pages and the pages are made up of objects, as shown in Figure 129." [graphically representable objects are combined into super-objects of higher complexity] Examples of combined graphically representable objects include graphics (p. 74) and bar codes (p. 85). [combined into super-objects of higher complexity before being stored in the memory.] At p. 38 – defined by a data object, 214, (super object of higher complexity – bar code comprised of individual fields is stored as a data object) At p. 40, "…create complex electronic output [super objects of higher complexity] combining variable data, overlays, bar coding, images, and other document elements." At p. 99, "overlay objects" represent a super object of higher complexity that may include vertical, horizontal, diagonal rules (lines), with different weights and thickness, boxed with and without shading, grids, arcs, and polygons, graphics or image, bar codes, text.

Regarding claims **3, 24, and 40**, IBM teaches (pp. 16-17 & 26, controls page development, job processing, job status and error recovery) feedback error messages recognizing a transformed graphic object which cannot be output by the printer. IBM teaches (p. 96) splitting the graphic object into part objects of lower complexity suitable for the output print data stream of the output device.

Regarding claims **4, 25, and 41**, IBM teaches (p. 404 –envelope stuffer, binder, stapler; p. 334-Z fold script) scripts controlling external devices [a graphically representable object stored in memory in object-oriented format is assigned a script which controls external devices]. IBM (p. 42, 29) teaches an archiving device and security equipment [controls access].

Regarding **claims 5 and 26,** IBM teaches (p. 332) an "overlay" keyword that specifies the inclusion of an overlay to be printed. (p. 276), Remote overlays are automatically received, "...received as an AFP resource without further conversion." [graphically representable object, stored in memory, assigned a script which automatically receives data] IBM teaches (p. 7) an object oriented system including electronic document elements such as overlays, fonts, and images.

Regarding **claims 17 and 20**, IBM teaches (p. 1) an AS/400 (Application System/400 with operating system OS/400, p. 400/data processing unit with memory) using an AFP printing and presentation system to transform digital print data streams, to communicate with printers through the Intelligent Print Data Stream (IPDS). The AS/400 (p. 28, interfaces stored in OS) and the printer devices (p. 38) both are data processing units with memory [a system for the transformation of digital print data streams comprising data processing unit having memory and communications interface, programmed to operate the method of claim 1]. IBM teaches (p. 1)

system management to communicate via twinax or to be network-attached via TCP/IP [the communications interface]. See IBM, p. 405, "Systems Network Architecture" (SNA) and "TCP/IP." See additional limitations of claim 17 addressed in claim 1 rejection above.

Per **claim 21,** IBM teaches (p. 27, 264) LAN print servers that manage print job files to either IPDS or ASCII printers.

New **claim 54** is very similar to claims 1 and 17 is rejected for the same reasons. IBM teaches a printer at 270.

Per **claims 56 and 62** (data processing unit is further programmed to cause original print and image data to be printed without a printer driver), IBM teaches: "HPT converts the AFP data stream into HP PCL or Lexmark PPDS print data streams. This transform works in one of two ways: raster or mapping. Raster mode builds an image of the page in AS/400 memory to send to the printer .... "(IBM at 270) Building the "image of the page" in raster mode shows the printing of a high-level print data stream (PCL or PPDS) into a low-level bitmap image (the "image of the page" in raster mode) for printing on a less-capable printer that does not have a PCL or PPDS printer driver specifically installed.

Claim 59 is very similar to issued claims 1, 17 and 20, and for the same reasons. The parser is specifically recited to "analyze syntax" of the print data stream. See discussion above in claim interpretation and Requester Comments09/06/2011, pp. 25-26.

Claim 61 is very similar to issued claims 1 and 5 and is rejected for the same reasons. The claim more generally refers to obtaining information over a "network." IBM teaches obtaining information over the network at pp. 1 and 405: "...to communicate via twinax or to be network-attached via TCP/IP...", "Systems Network Architecture" (SNA) and "TCP/IP."

Claim 68 is rejected for the same reasons as claims 1, 17, 21, 22, and 38 above and additionally specifically recites "analyze syntax." See this addressed in claim 59 above and in the claim interpretation section.

Claim 70 is rejected for the same reasons as claims 1, 17, 212, 22, 38 and 61 above.

Per **Claim 71** (output device is a printer), IBM teaches (270) "...builds an image of the page in AS/400 memory to send to the printer .... " (IBM at 270)

Per **Claim 76** (input data stream is formatted in a page description language), IBM teaches (pp. 1, 8; postscript) page description language.

Per claim 77 (performing syntactic analysis on said input data stream), see rejection of claim 59 above and the claim interpretation section. IBM teaches: "Network applications are generating document data streams, such as Postscript, and image formats, such as GIF and TIFF. Data in these formats can be converted to AFP and then stored or printed from the AS/400." (IBM at 8, emphasis added.) As stated by the PO: A syntax analyzer "permits the parser to analyze even complex page description languages (PDLs)." (PO Remarks 03/30/2011,p. 7, ln. 2). Postscript is a Print Description Language (PDL) that IBM uses as an input print data stream for conversion to the object-oriented AFP format. Therefore, by the PO's own definitions, IBM teaches the use of a syntax analyzer.

Per **claim 78**, IBM teaches class hierarchy. (IBM at 5-7, 23, and FIG. 5) See Third Party Comments, 09/06/2011, pgs. 5-7 with respect to IBM's use of objects with class hierarchy.

Per **claims 80-82** (objects are managed by display list management module / display list management supports one page and multi-page documents at a plurality of levels/ display list management can be expanded dynamically by new objects), IBM teaches (p. 39) an APU as a display list management module: "APU is an interactive, end-user system for transforming existing application output to advanced electronic output. Current line-mode output can be migrated to electronic business documents utilizing electronic forms, image, graphics, fonts, and bar codes. APU will create complex output, including multiple page formats and conditional logic, all without changes to the existing application." Using the APU for "transforming existing application output" (an input print data stream) to "complex output, including multiple page formats and conditional logic" shows the management of the objects by the display list management module as recited by the claims. IBM teaches (p. 161) the availability of the Advanced Print Utility (APU), which supports expansion by new objects: "APU, designed to build complex documents, has a simple end-user interface; documents can include such elements as multiple page formats, multiple copies, and standard back overlays. APU provides application data remapping, enabling you to completely change the way the application data is printed. APU also provides output conditional processing,
where data in the existing spooled output is used to determine document layout and flow." Using the APU display list management module to "completely change the way the application data is printed" including "multiple formats" and multiple copies" shows that the display list management can be expanded dynamically by new objects (new formats, new overlays, etc.) as recited in the claim.

Ground #2: Claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, 48-49, 54, 56, 59, 61-62, 68, 70-71, and 76-82 are rejected as obvious over IBM in view of Interleaf under 35 U.S.C. 103(a). Request 07/16/2010, pp. 5 & 7-8 and Exhibit H claim chart are incorporated by reference. See Requester Comments 09/06/2011, p. 21, regarding Ground #2, which are incorporated by reference.

IBM discloses the limitations of independent **claims 1, 17, 22, and 38** as noted above. To the extent that IBM fails to expressly teach "at least one stored script is assigned, which is executed" to the graphically representable objects, Interleaf discloses (p. 79) a "Lisp script... stored within or external to a document." The scripts are "attached to any Interleaf object" [stored script is assigned to the graphically representable objects] "A Lisp program generates the photo album document from the above data. Each generated page contains a frame with a face image, a text description about that person, and a button component labeled with the user's workstation name. The face frames and the button component objects are subclassed, with new select methods provided for each. Clicking on the face plays the audio name, and clicking in the button component produces a list of processes being run on the remote workstation." (Interleaf, p. 79) [script is assigned, which is executed in the cases defined in the script]

Therefore it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to modify the teachings of IBM with the teachings of Interleaf to expressly teach a script assigned to an object to be executed. The prior art references are analogous as both address object-oriented document publishing systems with scripting functionality. The combination of prior art elements is done according to known methods to yield predictable results. See MPEP 2141 .III.(A). Combining the printing process of IBM that parses and

transforms a print data stream into an object-oriented format, and the print-scripting process of Interleaf are combining well known printing techniques to yield predictable results. See MPEP 2141.III.(C), (D). Interleaf suggests the combination because it is explicitly designed to leverage outside printing and publishing functionality: "External publishing functionality. In addition to having the presentation and editing functionality within the document, document system based applications can also take advantage of all the other publishing facilities that exist within the document system. For example, the IMA application makes use of the system's printing and filtering capabilities." (Interleaf, p. 82) IBM provides external printing and publishing functionality: "Advanced Function Presentation (AFP) is a printing and presentation system that enables you to take full advantage of printing technology on your AS/400 system." IBM also teaches the desire to control outside printing and publishing functionality. "[P]ostprocessing option. A hardware device that attaches to the output side of a printer; for example, an envelope stuffer, binder, or stapler." (IBM, p. 404) A person of ordinary skill in the art would be motivated to "take full advantage" of the printing and publishing capabilities of the IBM system by combining it with the Interleaf system. See MPEP 141.III.(G).

Regarding **claims 2, 23, and 39**, in addition to the above noted teachings of IBM, Interleaf analogously teaches graphically-representable objects combined into super-objects of higher complexity: "Frames are containers used explicitly to control the placement of content including text and graphic objects such as diagrams, images, charts, equations, etc." (Interleaf, p. 76)

Interleaf also teaches (p. 78-79) a document as one example of a super-object of higher complexity. Having the component objects be changed "while the document is opening" and then

having the document (as a whole) be changed "back to the default document class.., in a state suitable for user completion" shows that the component objects being activated and combined into the document (the super-object of higher complexity). The technical process that "purges the localization table" and "changes the class of the document" so that the user can work with the document as a "default document" shows that the objects are combined into the super object of higher complexity before they are stored in the memory.

Regarding **claims 3, 24, and 40**, see the IBM teachings as noted above.

Regarding **claims 4, 25, and 41**, to the extent that the IBM teachings as noted above fail to expressly teach the claim limitations, Interleaf further teaches "Interleaf's active document architecture is described by English et al. [14], who define active documents as "structured documents and their processors in which the objects in the documents can be acted upon by, and can themselves act upon, other objects in the document or the outside world." (Interleaf, p. 78) [the script "controls external devices"/ external workstation] "The face frames and the button component objects are subclassed, with new select methods provided for each. Clicking on the face plays the audio name, and clicking in the button component produces a list of processes being run on the remote workstation." (Interleaf, p. 79)

Regarding **claims 5, 26, and 42**, to the extent that IBM fails to expressly teach a script which automatically receives data, Interleaf further teaches (p. 79) data preferably data organized in an object oriented manner by a LISP program…image data, text data… See also Interleaf "auto-localizing templates (p. 77-78). "The template documents contain tables of localization information, including the names of all components as well as values for localized variables such as page size .... A template document subclass is defined in one of the system Lisp libraries.

This subclass has a custom open method that changes component names [text data] and sets other document properties according to the table-specified settings for the current language."

Regarding **claims 6, 27, and 43**, the teachings are noted in the above anticipation rejection. Interleaf further expressly teaches (p. 86) that the script "requests this data automatically." Referring to the auto-localizing templates example discussed relative to claim 5: "Active documents need not appear active to all users (e.g., the auto-localizing templates)." Having the active document scripts change the document (in this case auto-localize the document) without the user knowing (the document does not appear to be active) shows that the scripts requested and received the localization data automatically as recited in the claim.

Regarding **claims 7, 28, and 44**, IBM teaches graphic objects and scripts, while Interleaf further expressly teaches (p. 78-79) that the script "reassigns the data received by it to the graphic object.., associated with itself": "A template document subclass is defined in one of the system Lisp libraries. This subclass has a custom open method that changes component names and sets other document properties according to the table-specified settings for the current language. This occurs while the document is opening, but before its content is shown. The open method then purges the localization table, and changes the class of the document back to the default document class, thus permanently deactivating this newly created document instance. This leaves its initial content in a state suitable for user completion." The document (and document text strings) are graphic objects as recited in the claim. Having the script "change the class of the document back to the default document class" after all the changes are made shows the script reassigning the data received by it to the graphic object associated with itself (the document). Interleaf teaches (p. 84) that the method "prints out the graphic object [the

document]... assigned to itself, together with the data requested." Interleaf further teaches: "If a document contains active objects within the document file stream (as opposed to the methods file), these become activated when the document is opened programmatically or by the user for viewing, editing, or printing." [printing out the graphic object /the document]

Regarding **claims 11, 32, and 48**, where IBM teaches graphically representable objects, stored in memory in the object oriented format and assigned at least one script. Interleaf further expressly teaches the script "is executed in the case of the output of the object defined in the script." Similar to the '789 patent, 6:6-16, Interleaf teaches (p. 84) event-based activation of scripts: "If a document contains active objects within the document file stream (as opposed to the methods file), these become activated when the document is opened programmatically or by the user for viewing, editing, or printing."

Regarding **claims 12, 33, and 49**, IBM teaches the graphically representable object stored in memory in an object oriented format is assigned at least one script. See teachings above. Interleaf further expressly teaches (78-79) assigning "at least one case relating to the execution of the script": A template document subclass is defined...has a custom open method that changes component names and sets other document properties according to the table-specified settings for the current language. This occurs while the document is opening, but before its content is shown. Defining a "custom open method" that runs "while the document is opening, but before its content is shown" shows defining at least one case relating to the execution of the script. Interleaf also teaches (p. 78-79) that the script is "occurring automatically, preferably without further influence from outside."

Regarding **claims 20 and 21**, as noted above, IBM teaches a printer / printer server characterized in that it has a system for the transformation of digital print data streams. Also see IBM definitions at p. 404.

Claim 54 recites limitation similar to claim 17/1 and is rejected for the same reasons under the combination of IBM and Interleaf. Claim 54 specifically outputs to a printer. IBM teaches (p. 270) a printer.

Claims 56, **59, 61-62, 68, 70-71, and 76-78 and 80-82** are rejected under the combination of IBM in view of Interleaf, for the reasons stated above (in anticipation rejection, as anticipated by IBM under 102(b)).

Regarding **claim 79**, IBM teaches (pp. 5-7) the use of objects with class hierarchy, but fails to explicitly teach "dynamically linking a plurality of objects." Interleaf teaches (p. 77), "The heart of I6 is its object system, which has a dynamic class hierarchy, message set, and method associations per class. The object system includes features such as multiple inheritance, property inheritance, and before and after methods [1]." Interleaf's "dynamic class hierarchy" shows the storing of graphically representable objects "based on membership in hierarchically organized classes." Interleaf teaches: "Lisp was chosen as the I6 extension language because it is interpreted (providing a fast prototyping environment) and has run-time binding. Run-time binding

is required for easy delivery of active documents. (Interleaf at 77). Interleaf's "run-time binding" shows the dynamic linking of a plurality of objects as recited in the claim.

Ground #3: Claims 1-14, 16-18, 20-35, 37-51, 53, 54, 56, 57, 59-62, 64-66, 68-74, and 76-82 are rejected under 35 U.S.C. 103(a) as obvious over IBM in view of Interleaf, and further in view of the Interleaf Patent. Request 07/16/2010 at pp. 10-12 and Exhibit I claim chart are incorporated by reference. (It is presumed that the Request, p. 12 has a typo in Proposed Rejection #3, and should include claim 53, and not include claim 52. This is consistent with the claim chart rejections in Exhibit I. Examiner has removed claim 52 from the list of rejected claims and inserted claim 53.) See Requester Comments 09/06/2011, pp. 21-22, regarding Ground #3, which are incorporated by reference.

Per **claims 1, 17, 22, and 38,** IBM / Interleaf disclose teachings as noted above. Interleaf Patent further expressly teaches (1:30-41) there is "at least one stored script is assigned" to the graphically representable objects, "We have discovered an improved document processing system for creating computer procedures (i.e., methods) and associating them with an electronic document structured as a nested hierarchy of constituent objects. The system includes a document processing means for defining the method and associating it with the electronic document or its lower level constituent objects. The system further includes an interpreter for interpreting the computer procedure in response to a specified event. In preferred embodiments, the methods are defined in a dialect of the programming language LISP." The computer procedures "defined... in LISP" are the stored scripts as recited in the claim. "Associating [a computer procedure] with the electronic document or its lower level constituent objects" shows that at least one stored script is assigned to the graphically representable objects as recited in the claim. The Interleaf Patent also teaches (3:37-39) that the script "is executed in the cases defined in the script", "As explained above, any data object within the hierarchy may be

associated with a method which, upon the occurrence of a certain event, will be implemented." "Implement[ing]" the method "upon the occurrence of a certain event" shows the script being executed in the cases defined in the script as recited in the claim. Interleaf continues (6:32-56), "In defining a new method, the user must also select the event which will trigger the execution of the new method. The following are examples of events which can be used to trigger methods (Note: certain events are available as triggers to only certain classes of objects): 1. the opening or closing of a document object, 2. the saving of a document, or automatically saving a backup version, 3. the user entering a particular content in the document associated with the procedure, 4. the user entering a particular content in the different document, 5. the starting of an external application, 6. an application outputting a particular result, 7. the mouse being moved to a certain position, 8. any test or graphic object being selected, 9. any popup menu choice of user interface 14 affecting any text or graphic object, 10. the system clock reaching a certain time or date, 11. any mouse button being pressed, 12. any key stroke or set of keystrokes, and 13. an attempt to edit any text of graphic object." The events "which will trigger the execution of the new method" are the cases defined in the script as recited in the script. Executing the script in response to the cases shows the script being executed in the cases defined in the script as recited in the claim.

Reasons to Combine IBM, Interleaf and the Interleaf Patent

Therefore, it would have been obvious, to one of ordinary skill in the art, at the time of the invention, to combine the teachings of IBM, Interleaf, and the Interleaf Patent for the following reasons: 1) A person of ordinary skill in the art would have reason to combine IBM

and Interleaf because they address the same field, namely object-oriented document publishing systems with scripting functionality. The above-described combination of prior art elements is done according to known methods to yield predictable results. See MPEP 2141.III.(A). 2) Combining the printing process of IBM that parses and transforms a print data stream into an object-oriented format, and the print-scripting process of Interleaf and the Interleaf patent are combining well known printing techniques to yield predictable results. See MPEP 2141.III.(C), (D). 3) Interleaf suggests the combination because it is explicitly designed to leverage outside printing and publishing functionality: "External publishing functionality. In addition to having the presentation and editing functionality within the document, document system based applications can also take advantage of all the other publishing facilities that exist within the document system. For example, the IMA application makes use of the system's printing and filtering capabilities." (Interleaf, p.82) IBM provides external printing and publishing functionality: "Advanced Function Presentation (AFP) is a printing and presentation system that enables you to take full advantage of printing technology on your AS/400 system." IBM also teaches the desire to control outside printing and publishing functionality. "[P]ostprocessing option. A hardware device that attaches to the output side of a printer; for example, an envelope stuffer, binder, or stapler." (IBM, p. 404) A person of ordinary skill in the art would be motivated to "take full advantage" of the printing and publishing capabilities of the IBM system by combining it with the Interleaf system. See MPEP 2141.III.(G). 4) Interleaf and the Interleaf Patent are both directed to the same physical system. The existence of this system serves as an explicit suggestion to combine the teachings of Interleaf and the Interleaf Patent, which both describe the system. This would motivate a person of ordinary skill in the art to combine

Interleaf with the Interleaf Patent to gain more information about the underlying product. 5) A person investigating the Interleaf reference would be motivated by "common sense" to find other descriptions of the same underlying functionality. This "common sense" would motivate a person of ordinary skill in the art to combine Interleaf and the Interleaf Patent.

Regarding **claims 2, 23, and 39**, IBM and Interleaf disclose limitations as noted above. Expressly Interleaf Patent teaches (2:61-3:13) that the graphically-representable objects "are combined into super-objects of higher complexity": "DPS 28 structures documents as a nested hierarchy of data objects." See FIGS. 3 and 5, an author object 34 identifying the author of the memorandum, a destination object 36 identifying the person to whom the memorandum is addressed, and a topic object 38 summarizing the subject of the memorandum. "Each object 34-48 may consist of a plurality of lower level objects. For example, each word or character of paragraph 40 may be treated as a separate object and each symbol representing a logic gate 50-54 of drawing 48 may be treated as a distinct object."

Regarding **claims 3, 24, and 40**, IBM and Interleaf disclose limitations as noted above.

Regarding **claims 4, 25, and 41**, IBM and Interleaf disclose limitations as noted above.

In addition, the Interleaf Patent also teaches that the script "controls external devices." Interleaf Patent (10: 16-20) recites, "The electronic document processing system of claim 2 further comprising a network interface means for interfacing with a network and for enabling said interpreter to execute the procedures by executing subprocedures stored on at least one remote device connected to the network." (Interleaf Patent, claim 9). Having the script "execut[e] subprocedures stored on at least one remote device" shows the script controlling external devices

as recited in the claim. The Interleaf Patent also shows additional examples where the script controls a frame grabber and phone system / controls external devices." A customer receipt that automatically runs a "frame grabber" and incorporates a photographic image of the purchaser." (Interleaf Patent, 7:47-49) A phone directory that can call the person who's name you've selected; if the line is busy, it puts you into electronic mail." (Interleaf Patent, 8:27-30)

Regarding **claims 5, 26, and 42**, IBM / Interleaf provide an obvious teaching, as noted above, and Interleaf Patent further expressly teaches (7:3-7), a "report on how a programming project is proceeding which automatically imports data from the computer-aided software engineering (CASE) application. The report has hyperlinks to the CASE software automatically installed." [script "automatically receives data] Other examples of the script automatically receiving the data include, "A customer receipt that automatically runs a "frame grabber" and incorporates a photographic image of the purchaser." [organized as image data] (Interleaf Patent, 7:47-49); "A stock-market information center that retrieves stock information and displays it in text and charts that can be automatically distributed or incorporated in other documents." [organized as text data] (Interleaf Patent, 7:53-55); "A weekly report that builds a time-line based upon information in a corporate database. If anyone is more than a week behind, it queries the database for more information and builds a document on a manager's desktop." (Interleaf Patent, 8:10-14). See additional examples in Exhibit I.

Regarding **claims 6, 27, and 43,** IBM / Interleaf provide an obvious teaching, as noted above, and Interleaf Patent further expressly teaches examples that show that the script "automatically receives data" and "requests this data automatically." Interleaf Patent, 7:3-7, a

report automatically imports, using hyperlinks, data from a computer-aided software engineering (CASE) application. (Interleaf Patent, 7:3-7) See additional examples in Exhibit I.

Per **claims 7, 28, and 44**, the teachings of the Interleaf Patent further support the combined teachings of IBM / Interleaf as noted in the previous rejection combination. As an example of the script "reassigns the data received by it to the graphic object...associated with itself", "A customer receipt that automatically runs a "frame grabber" and incorporates a photographic image of the purchaser." (Interleaf Patent, 7:47-49) Putting the image into the customer receipt shows that the data is reassigned to the graphic object (it incorporates the data in the text and charts) as recited in the claim. See additional examples in Exhibit I.

Per claims **8, 29, and 45**, the IBM/Interleaf combination discloses teachings as noted above in the rejection of claim 1. Additionally Interleaf Patent expressly teaches examples that show that the script "automatically sends data": "An urgent memo that integrates with voice annotation software so when it mails itself [script automatically sending data] to someone, it announces its presence audibly." (Interleaf Patent, 7:3-7) or "A document created from a database which updates the database [script automatically sending data] if you make any changes in the imported document." (Interleaf Patent, 7:47-49) See additional examples in Exhibit I.

Per **claims 9, 30, and 46**, the IBM / Interleaf combination discloses teachings as noted above. Additionally Interleaf Patent expressly teaches "the graphic object associated with the script is sent to a receiver": "A WYSIWYG document that can send itself over an electronic mail system." (Interleaf Patent, 7:50-51) The email recipient is the receiver. Having the

document send itself over an electronic mail system shows the graphic object associated with the script (the document) is sent to a receiver. See additional examples in Exhibit I.

Per **claims 10, 31, and 47**, the IBM / Interleaf combination discloses teachings as noted above. Additionally Interleaf Patent expressly teaches that the script "reassigns the data received by it to the graphic object...associated with itself." As an example, "A stock-market information center that retrieves stock information [script automatically receiving data] and displays it in text and charts that can be automatically distributed or incorporated in other documents." (Interleaf Patent, 7:53-55) See additional examples in Exhibit I.

Per **claims 11, 32, and 48**, the IBM / Interleaf combination discloses teachings as noted above. Additionally Interleaf Patent expressly teaches the script "is executed in the case of the output of the object defined in the script." As described in the '789 patent (('789 patent, 6:6-16) being executed in the case of the output of the object defined in the script refers to event-based activation of the script. Interleaf similarly teaches event-based activation of scripts at p. 84. Scripts that are activated "when the document is opened programmatically [for] printing" corresponds to the "ON-PRINT" activation of scripts described in the specification of the '789 patent. The Interleaf Patent also teaches this claim feature: "As explained above, any data object within the hierarchy may be associated with a method [script] which, upon the occurrence of a certain event, will be implemented." (Interleaf Patent, 3:37-39) "In defining a new method, the user must also select the event which will trigger the execution of the new method..." (Interleaf Patent, 6:32-56)

Per claims **12, 33, and 49**, the IBM / Interleaf combination discloses assigning "at least one case relating to the execution of the script, as noted above. (Interleaf, pp. 78-79 – custom open method that changes component names and sets other document properties according to the table-specified settings for the current language, occurring automatically.) The Interleaf Patent also teaches (6:32-33) assigning "at least one case relating to the execution of the script": "In defining a new method, the user must also select the event [assigning "at least one case relating to the execution of the script] which will trigger the execution of the new method." The Interleaf Patent also teaches that the script is "occurring automatically, preferably without further influence from outside." As an example, Interleaf Patent teaches (7:47-49), "A document created from a database which updates the database if you make any changes in the imported document." The script is executing automatically, without any further influence from outside as recited in the claim. See other examples at Exhibit I.

Per **claims 13, 14, 34, 35, 50 and 51**, the IBM / Interleaf combination discloses a script defining an automatically occurring case. IBM/Interleaf fail to teach the script "is configured as a timer, that is to say as a case which occurs automatically as a result of expiry of time." Interleaf Patent discloses (6: 32-53) the system clock reaching a certain time or date. For example (Interleaf Patent, 7:8-10), "A press release that knows to print "Draft" across its pages until the system clock reaches the announcement date listed at the beginning of the text." As another example Interleaf Patent teaches (7:31-34), "A time-critical document that knows when it's due and appears on screen to remind you of when you need to work on it." The Interleaf Patent teaches (8:10-14) "A weekly report that builds a time-line based upon information in a corporate database. If anyone is more than a week behind, it queries the database for more

information and builds a document on a manager's desktop." [timer operates cyclically, that is to say it starts itself again upon expiry]

Regarding **claims 16, 37, and 53,** Interleaf Patent (2:34-36) teaches, "In addition to modifying the appearance of a document, a user may define a procedure [a script object] to be associated with the document." The Interleaf Patent teaches manipulation of the objects "preferably before they are output in the output print stream": "Electronic Document Processing Systems generally include a computer workstation programmed to allow a user to create and edit an electronic representation of a document. The workstation includes a display device for displaying an image of the document as it would appear [before it is output] if printed." (Interleaf Patent, 1; 15- 20) The Interleaf Patent teaches (1: 15-27), "Electronic Document Processing Systems [application interface] generally include a computer workstation programmed to allow a user to create and edit an electronic representation of a document. The workstation includes a display device for displaying an image of the document as it would appear if printed. The system allows the user to edit the document by modifying existing text and graphics or by adding entirely new text or images. The Interleaf Patent teaches (1: 15-27) that the application interface allows the graphically representable objects "to be read out, to be changed, to be deleted or to have new objects (5) appended."

Per **claim 18**, the Interleaf Patent teaches (1:15-27) the system computer workstation [operating station] programmed to allow a user to create and edit an electronic representation of a document. The Interleaf Patent teaches (2: 19-25) manipulating a keyboard 16, a mouse 18 [input means], using a processor 20 and display 14 [display means] with image data representative of the selected document. In response, display 14 generates an image of the

document for viewing by the user." Interleaf Patent (1:18-27) teaches, "The workstation includes a display device for displaying an image of the document as it would appear if printed. For example, a document such as a magazine article may contain an arrangement of text and graphics. The system allows the user to edit the document [read out via application interface and change] by modifying existing text and graphics or by adding entirely new text or images. As the user modifies the document, the display is continually updated to reflect these changes, thereby allowing the user to interact with the system to achieve the desired document."

Per **claims 20 and 21**, IBM teaches a printer and printer server as noted in the prior rejections.

Claim 54 is rejected as obvious under the combination of IBM, Interleaf, and Interleaf Patent for the same reasons given above as related to system claim 17/1. Claim 54 specifically outputs to a printer. IBM teaches (270) a printer.

Claims 56, 59, 61-62, 68, 70-71, and 76-78 and 80-82 are rejected under the combination of IBM in view of Interleaf, in view of Interleaf Patent, for the reasons stated above (in anticipation rejection, as anticipated by IBM under 102(b)). Additionally, the rejection of claims 80-82 are further supported by Interleaf teachings (pp. 76 & 83, "editor objects as display list management modules)

Claims 57 recites limitations similar to claims 17 and 18 and is rejected for the same reasons.

Claim 60 recites limitations similar to claims 1, 13-14 and is rejected for the same reasons.

Claim 64 recites limitations similar to claims 1, 17, and 18 and is rejected for the same reasons. Claim 64 also recites "analyze syntax" of the data stream. See IBM. p. 3. A syntax analyzer "permits the parser to analyze even complex page description languages (PDLs)." (PO Remarks 03/30/3011 at p. 7). Postscript is a Print Description Language (PDL) that IBM uses as an input print data stream for conversion to the object-oriented AFP format. Therefore, by the PO's own definitions, IBM teaches the use of a syntax analyzer. Also see "analyze syntax" taught by Interleaf at p. 16.

Per **claim 65**, see limitations addressed in the rejection of claims 1 & 18 above.

Claim 66 is rejected under the combination of IBM in view of Interleaf, in view of Interleaf Patent, for the reasons stated above in the rejection of claim 62. (in anticipation rejection, as anticipated by IBM under 102(b)).

Claim 69 recites limitations similar to claims 13-14 above and is rejected for the same reasons.

Claim 72 recites limitations similar to claim s 1, 17, 18, 20, and 21 and is rejected for the same reasons. Claim 72 includes the term "analyze syntax." The limitation is addressed in the rejection of claim 64 above.

Claim 73 recites limitations similar to claim 65 and is rejected for the same reasons.

Claim 74 is rejected under the combination of IBM in view of Interleaf, in view of Interleaf Patent, for the reasons stated above in the rejection of claim 62. (in anticipation rejection, as anticipated by IBM under 102(b)).

Claim 79 is rejected for the same reasons given in the obvious rejection (IBM in view of Interleaf) above.

Ground #4: Claims 15, 19, 36, and 52 are rejected under 35 U.S.C. 103(a) as obvious over IBM in view of Interleaf and further in view of the Interleaf Patent and Lieberman. The Request (07/16/2010) at pp. 10-13 and Exhibit I claim chart are incorporated by reference.

Per **claims 15, 36, and 52**, Lieberman supplies the missing teaching of the IBM/Interleaf/ Interleaf Patent combination. Lieberman teaches (p. 17) that Javascript is a common formal language for scripts. "The intent of these efforts is primarily to support inter-application data-sharing, scripting (manually writing external programs to control an application)...Examples of such efforts include … Javascript, and others…"

Reasons to Combine IBM, Interleaf, the Interleaf Patent, and Lieberman

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of IBM, Interleaf, Interleaf Patent and Lieberman for at least the following reasons: 1) Java Script, as taught by Lieberman, the specific language used in scripting is "merely a matter of obvious engineering choice," and thus prima facie obvious. (MPEP § 2141.04.V.(B)). The '789 patent suggests no unexpected or unpredicted benefit of using Java Script in this manner, and in fact merely refers to Java Script as one of several "example" scripting languages that could be used. ('789 patent, 6:10-11). Thus, Lieberman teaches that substituting Lisp (as taught in Interleaf) for Javascript is just substituting one known solution (Lisp) for another known solution (Javascript). A person of ordinary skill in the art would be motivated to combine Interleaf with the Interleaf Patent and Lieberman because they both address the same field, namely scripting interfaces for object systems. This overlap in the

field of use would motivate a person of ordinary skill in the art to combine Interleaf, the Interleaf Patent, and Lieberman.

Per **claim 19**, Interleaf Patent teaches (1:18-27; 5:38-41), "The workstation includes a display device for displaying an image of the document as it would appear if printed. For example, a document such as a magazine article may contain an arrangement of text and graphics. The system allows the user to edit the document [objects to be read out, changed, deleted, appended]. The combination of IBM/ Interleaf/ Interleaf Patent modified with Lieberman's suggestion of "preferably also Java Script objects (5a) themselves" can be read or changed. See claim 15 above.

Ground #5: Claims 1-2, 4-7, 11-12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 are rejected under 35 USC 102(b) as being anticipated by Interleaf under 35 U.S.C. 102(b). The Request (07/16/2010) at pp. 10-14 and Exhibit I claim chart are incorporated by reference.

Per **claims 1, 22, and 38**, Interleaf teaches (p. 75, 80) a document processing system that uses "active documents", which are documents "built with an extensible object system." (Interleaf, p. 75) and includes document system facilities for printing, filtering to accept CALS compliant [a government standard] documents [input data streams], and automatic hypertext linking and indexing." Interleaf describes (p. 84) receiving a "document file stream" for printing, and parses the document to see if it has any active objects: "If a document contains active objects within the document file stream ... these become activated when the document is opened programmatically [read in input data stream/ analyzed by means of a parser/ LISP interpreter @

p. 77] or by the user for ... viewing, editing, printing." Interleaf teaches (p. 76) changing the style "in the middle of a component text stream." [a method for transformation of a digital print data stream] Interleaf provides another example of a method for transformation of a digital print data stream (p. 81-84) by changing the objects in the stream or by programmatically apply different graphic properties or transformations to the object, such as changing its color or size or position. Interleaf teaches (p. 84) saving code of the document itself, either within the main document file stream or in an auxiliary file containing method definitions. [digital print data streams] Interleaf teaches (p. 77, 85) "Lisp is a good language for active document development due to features such as the interpreter (analyzed by parser)…functions as data… and run-time binding…" The "run-time binding" of the "active document objects" by the Lisp interpreter shows that the data stream is analyzed by means of a parser. Interleaf shows (p. 85) the parsing and analysis of the objects during activation: "The system contains two facilities to limit activation .... The second is to define an "active load hook" that gets called prior to each automatic activation. It is passed the current object and the code that is about to be evaluated. A sophisticated program can examine this code and decide whether or not to permit its activation. We note that examination of code by code is easily done in Lisp, which makes no distinction between code and data. A simple example of such an inspector is distributed with the system." When a document file stream is loaded, the "active load hook" examines [parser analyzing input document file] the code objects that are about to be activated. Interleaf teaches (p. 81-82) parsing the input print data stream "for graphically representable objects." Objects in the input print data stream are graphically representable objects: document objects to represent UI elements can be changed by applying different graphic properties or transformations to the

object, such as changing its color or size or position or hide / show abilities. [transformed into a format] See p. 83 for examples of document objects. Interleaf provides an example object called a "Named Graphic Object," or "NGO." (Interleaf, p. 81) Interleaf teaches storing (p. 7, 82) the graphically representable objects in a memory. Lisp has run-time binding (graphically representable objects in memory). Interleaf (p. 76, 82), "The system should be object-oriented [object-oriented format] to allow for easy active document building and reusability of active objects." [reuse objects stored in memory]

Interleaf teaches (p. 82) that the transformation capabilities are used to control publishing: A simple example is the ability to have mixed fonts and /or text properties in part of your UI. A more advanced example is the ability to do automatic pagination of floating graphic frames in a document with tables and images...easy to do with document-based solutions...In addition to having the presentation and editing functionality within the document, document system based applications can also take advantage of all the other publishing facilities that exist within the document system...the system's printing and filtering capabilities [transformed document is received at output device / preferably a printer]." The data stream that is sent to the system printing capability is the output print data stream. Opening the document for printing shows the system actually outputting the output print data stream.

Interleaf teaches (p. 77) "Lisp scripts can be attached to any Interleaf object, stored within or external to a document." [at least one stored script is assigned" to the graphically representable objects]

Interleaf teaches (p. 79) a photo album document including a button component. Clicking on the button activates a script. [script is executed in the cases defined in the script] Also see the “auto-localizing template” example cited in Exhibit J - a custom open method that changes component names and sets other document properties according to the table-specified settings for the current language. The "cases defined in the script" are when the document opens. As it executes, it "changes component names and sets other document properties according to the table-specified settings for the current language." This shows the script being executed in the cases defined in the script.

Per **claims 2, 23, and 39**, Interleaf teaches (p. 76) an object-oriented structured document editor, I6. The main structure is called a component, which contains content as well as formatting information. A “frame” container [super object of higher complexity] controls the placement of content including text and graphic objects such as diagrams, images, charts, equations, etc. Having a "frame" made up of including text and graphic objects such as diagrams, images, charts, equations [graphically representable objects] shows the combining of graphically-representable objects into super-objects of higher complexity. Interleaf teaches a template document subclass with (p. 78-79) an “open” method used for auto-localizing documents. The “open” method changes component objects in the localization table when a document [super-object of higher complexity] is opened and changes the class of the document so that the user can work with the document as a "default document." This example shows that objects are combined into the super object of higher complexity before they are stored in the memory. The “open” method then purges the localization table, and changes the class of the

document back to the default document class, thus permanently deactivating this newly created document instance and leaving its initial content in a state suitable for user completion.

Per **claims 4, 25, and 41** Interleaf teaches that the script "controls external devices." Interleaf teaches (p. 78) active documents as "structured documents and their processors in which the objects in the documents can be acted upon by, and can themselves act upon, other objects in the document or the outside world." Interleaf shows control of an external workstation. [external device] Interleaf teaches (p. 79) control of devices by way of button components. As an example, producing a "list of processes being run on the remote workstation" shows that the script controls external devices. The IMA (Intelligent Maintenance Aid) application example (p. 80-82) "makes use of the system's printing and filtering capabilities." Preferences stated in claim language do not constitute a limitation on the scope of the claim; the broadest reasonable interpretation is that other, non-preferable solutions are also encompassed in the claim scope. Broadly, archiving devices, folding systems, enveloping systems, and security equipment are within the scope of "objects in the outside world."

Per **claims 5, 26, and 42**, Interleaf teaches (p. 79) an example active document employee photo album that is programmatically generated using a LISP program (script). The LISP program receives data in the voice mail system, an image, and text from a database to generate the album. See other examples in Exhibit J.

Per **claims 6, 27, and 43**, Interleaf teaches the photo album example, as noted above. As an example (p. 79) the newly defined "frame" subclass contains face images [graphically representable object], includes a new "select" method [script], and opens/ plays the named audio

object. [automatically receives data, automatically requests data] See other examples in Exhibit J.

Per **claims 7, 28, and 44,** Interleaf teaches a custom open method that changes component names and sets other document properties [reassigns the data received] according to the table-specified settings for the current language. The document and document text strings are graphic objects. The script also changes the class of the document back to the default document class after all the changes are made [reassigning the data received by it to the graphic object associated with itself] Interleaf teaches (p. 84) that the method "prints out the graphic object.., assigned to itself, together with the data requested."

Per **claims 11, 32, and 48**, Interleaf discloses (p. 84), "If a document contains active objects within the document file stream (as opposed to the methods file), these become activated [executed in the case of the output of the object defined in the script] when the document is opened programmatically or by the user for viewing, editing, or printing." See '789, 6: 6-16.

Per **claims 12, 33, and 49**, Interleaf teaches (p. 78-79) "A template document subclass is defined in one of the system Lisp libraries. This subclass has a custom open method [defining at least one case relating to the execution of the script] that changes component names and sets other document properties according to the table-specified settings for the current language. This occurs while the document is opening [occurring automatically], but before its content is shown. The open method then purges the localization table, and changes the class of the document back to the default document class, thus permanently deactivating this newly created document instance. This leaves its initial content in a state suitable for user completion."

Ground #6: Claims 15, 36, and 52 are rejected under 35 USC 103(a) as obvious over Interleaf in view of Lieberman under 35 U.S.C. 103(a). The Request (07/16/2010) at pp. 13-15 and Exhibit J claim chart are incorporated by reference.

Regarding **claims 15, 36, and 52**, to the extent that Interleaf fails to expressly teach Java Script used as a format language for the scripts, Lieberman discloses (p. 17) Java script as a common formal language for scripts.

Reasons to Combine Interleaf and Lieberman

Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to combine Interleaf and Lieberman for at least the following reasons: 1) The specific language used in scripting is "merely a matter of obvious engineering choice," and thus prima facie obvious. (MPEP § 2141.04.V.(B)). The '789 patent suggests no unexpected or unpredicted benefit of using Java Script in this manner, and in fact merely refers to Java Script as one of several "example" scripting languages that could be used. ('789 patent, 6:10-11). Thus, Lieberman teaches that substituting Lisp (as taught in Interleaf) for Java script is just substituting one known solution (Lisp) for another known solution (Java script). 2) A person of ordinary skill in the art would be motivated to combine Interleaf and Lieberman because they both address the same field, namely scripting interfaces for object systems. This overlap in the field of use would motivate a person of ordinary skill in the art to combine Interleaf and Lieberman.

Ground #7: Claims 1-2, 4-14, 16-18, 22-23, 25-35, 37-39, 41-51, 53, 54, 57, 59, 61, 68, 70, and 76-82 are rejected as obvious over Interleaf in view of the Interleaf Patent under 35

U.S.C. 103(a). The Request (07/16/2010) at pp. 13-15 and Exhibit K claim chart are incorporated by reference. [Examiner presumes a typo and has removed claim 15 from the list of rejected claims. The 35 USC 103(a) rejection of Claim 15 as obvious over Interleaf in view of the Interleaf Patent is **not adopted** as proposed under Ground #7.] See Requester Comments 09/06/2011, p. 22, regarding Ground #7, which are incorporated by reference.

Regarding **claims 1, 17, 22, and 38**, Interleaf describes (p. 80, 84) a document processing system, active documents and an extensible objects system and to printing of such documents. Interleaf teaches (p. 76) a transformation (p. 81-82-transformations to the object) of the digital print data stream. See the Interleaf anticipation rejection above mapping citations to claim limitations.

To the extent that Interleaf fails to explicitly disclose claim limitations, the Interleaf Patent (1: 30-41) additionally teaches the transformation of digital print data streams. The "system includes a document processing means for defining the method and associating it with the electronic document or its lower level constituent object" [at least one stored script is assigned to the graphically representable objects] and an interpreter [parser]. Interleaf Patent discloses (3:37-39), "...any data object within the hierarchy may be associated with a method [script] which, upon the occurrence of a certain event, will be implemented." [script is executed in the cases defined in the script] "In defining a new method, the user must also select the event which will trigger the execution of the new method. The Interleaf Patent teaches a data processing unit / programmed computer workstation (FIG. 1A & 1: 15-18) including (2: 19-25; a

keyboard 16, mouse 18, processor 20, display 14, and memory 12. Interleaf Patent expressly teaches (2: 43-50) a computer readable medium to cause a processor to execute a method. Interleaf patent expressly teaches (FIG. 2 ; 2: 43-50) a "computer data signal embodied in a carrier wave and representing sequences of instructions", as the software used by EDPS within a network file system embodies sequences of instructions to be transmitted / embodied in a carrier wave.

Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention combine the teachings of Interleaf with the Interleaf for at least three reasons: 1) Interleaf and the Interleaf Patent are both directed to the same physical system. The existence of this system serves as an explicit suggestion to combine the teachings of Interleaf and the Interleaf Patent, which both describe the system. This would motivate a person of ordinary skill in the art to combine Interleaf with the Interleaf Patent to gain more information about the underlying product. 2) A person investigating the Interleaf reference would be motivated by "common sense" to find other descriptions of the same underlying functionality. This "common sense" would motivate a person of ordinary skill in the art to combine Interleaf and the Interleaf Patent. 3) A person of ordinary skill in the art would be motivated to combine Interleaf with the Interleaf Patent because they both address the same field, namely object-oriented document publishing systems with scripting functionality. This overlap in the field of use would motivate a person of ordinary skill in the art to combine Interleaf and the Interleaf Patent.

Regarding **claims 2, 23, and 39**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches (2: 61-3: 13) graphically representable objects

combined into super-objects of higher complexity, where each object may consist of a plurality of lower level objects.

Regarding **claims 4, 25, and 41**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches (10: 16-20) "...interpreter to execute the procedures by executing sub-procedures stored on at least one remote device connected to the network." [script controls external devices] See also Interleaf Patent 7: 47-49 and 8: 27-30.

Regarding **claims 5, 6, 26, 27, 42 and 43**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches examples that show the script "automatically receives data" / "requests data automatically." See Interleaf Patent 7: 3-7-report automatically imports data; 7: 47-49-automaticaly import a photographic image [automatically receiving data preferably organized as image data]; 7: 53-55- automatically insert retrieved stock data into reports; 8: 10-14-script automatically receives data from database. See more examples cited in Exhibit K.

Regarding **claims 7, 28, and 44**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches examples of the script reassigning "the data received by it to the graphic object...associated with it: 7: 47-49-placing/reassigning retrieved image into customer receipt; 7: 53-55-reassigning received stock data into text and charts; 8: 10-14-received database information reassigned to report graphic object. See more examples cited in Exhibit K.

Regarding **claims 8, 29, and 45**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches examples of the "script automatically sends

data": 7: 3-7-self mailing of memo with audible data; 7: 47-49-imported created document automatically updates database when changes are made [organized as text data object in document object]; 7: 50-51- a WYSIWYG document can send itself over an electronic mail system [organized as e-mail]. See more examples cited in Exhibit K.

Regarding **claims 9, 30, and 46**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches examples of the graphic object associated with the script is sent to a receiver: 7: 66-67-memos sent to "you" [receiver] as a reminder of dates / events; 8: 19-22-document [graphic object] remails itself to lawyer [receiver] upon changes after lawyer approval.

Regarding **claims 10, 31, and 47**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches the script reassigns the data received by it to the graphic object...associated with itself: 7: 53-55-stock market data reassigns data into text and charts graphic objects; 8: 10-14-incorporating database information into a report graphic object. See additional examples in Exhibit K.

Regarding **claims 11, 32, and 48,** Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches the script is executed in the case of the output of the object defined in the script: 3: 37-39-any data object within the hierarchy may be associated with a method [script] which, upon the occurrence of a certain event [executed in the cases defined in the script/script activated upon trigger event], will be implemented.

Regarding **claims 12, 13, 14, 33, 34, 35, 49, 50 and 51**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches examples of

"assigning at least one case relating to the execution of the script" / "timer...occurs automatically as a result of expiry of time": 6: 32-53-user selects the event which will trigger [occurring automatically] execution of new method / system clock reaching a certain time or date; 7: 47-49-script automatically updates database upon changes; 7: 66-67-script that sends memo to you automatically reminding of important dates and events; 8: 19-22-script re-sends lawyer approved document back to lawyer upon changes. The Interleaf Patent also teaches a timer with cyclical operation: 8: 10-14- weekly report builder queries database for more information. See additional examples in Exhibit K.

Regarding **claims 16, 37, and 53**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches examples of a script object: 2: 34-36-a user defined procedure. The Interleaf Patent discloses (1: 15-20) the document displayed "as it would appear if printed" [before output in an output print data stream] and the document is kept ready in (1: 15-27) an application interface, where the document is able to be edited, changed, etc.

Regarding **claim 18**, Interleaf teaches limitations as noted in the anticipation rejection above. The Interleaf Patent also teaches (FIG. 1A; 2: 10-25) a programmed computer workstation for an electronic document processing system, including display means (display 14) and input means (mouse 18 and keyboard 16). Interleaf Patent teaches (1: 18-27-display edit, modify document prior to output) graphically representable objects and script objects stored in memory are read out via the application interface to be changed, deleted, appended, preferably before they are output in the output print data stream.

Regarding **claim 54**, is rejected for the same reasons as claims 1, 17, 20, and 21 above. Claim 54 specifically recites output to a printer. Interleaf (p. 82) teaches "the system's printing and filtering capabilities." Further in support of Interleaf's teachings, Interleaf Patent teaches (10:16-20) teaches controlling external devices. Analyzing by a parser is discussed in claim interpretation section above. The Interleaf / Interleaf Patent fairly teaches analyzing by a parser.

Claims 57 and 58 recite limitations similar to claims 1, 16, 17, and 18 and are rejected for the same reasons. Interleaf teaches (p. 77) "Lisp scripts can be attached to any Interleaf object, stored within or external to a document." [stored script is assigned to graphically representable objects] More explicitly the Interleaf Patent teaches (FIG. 1A; 2: 10-25) a programmed computer

workstation (operating station) for an electronic document processing system, including display means (display 14) and input means (mouse 18 and keyboard 16). Interleaf Patent teaches (1:18-27-display edit, modify document prior to output) graphically representable objects and script objects stored in memory are read out via the application interface to be changed, deleted, appended, preferably before they are output in the output print data stream.

Claim 59 recites limitations similar to claims 1, 17, and 20 and is rejected for the same reasons. Claim 59 recites a parser to analyze syntax. Interleaf (p. 84) fairly discloses a parser that analyzes. See claim interpretation section above. Interleaf teaches (p. 77) "Lisp scripts can be attached to any Interleaf object, stored within or external to a document." [stored script is assigned to graphically representable objects] Interleaf (p. 82) teaches "the system's printing and filtering capabilities."

Claim 61 recites limitations similar to claims 1 and 5 and is rejected for the same reasons.

Claim 68 recites limitations similar to claims 1, 17, 18, and 64 and is rejected for the same reasons. The prior discussion related to "analyze syntax" applies.

Claim 70 recites limitations similar to claims 1, 5 and 61 above and is rejected for the same reasons.

Claim 76 (input data stream is formatted in a page description language) is disclosed by Interleaf (p. 76): "Frames are containers used explicitly to control the placement of content including text and graphic objects such as diagrams, images, charts, equations, etc. The author creates a frame within a component. One of the formatting properties of a frame controls its location relative to its token (creation point) within its parent component. Expressing frame location relative to its anchor token allows greater layout flexibility than requiring the user to place the frame on a specific place on a specific page." Describing the location of page elements programmatically, rather than as a series of points in a page image, shows that the input representation loaded by Interleaf uses a page description language.

Claim 77 relates to a syntax analyzer...analyzing by means of said parser...performing syntactic analysis.... Interleaf teaches that the Lisp environment includes "an editor, listener, compiler, debugger, syntax checker, help system a number of other utilities, and the entire source set for the Lisp portion of the system." (Interleaf at 77; emphasis added.) Including a "syntax checker" shows that integration of a syntax analyzer as part of the Interleaf system. The Interleaf interpreter includes a parser that performs syntactic analysis.

Claim 78 (storing graphically representable objects based on membership in hierarchically organized classes) is disclosed by Interleaf (p. 77): "The heart of I6 is its object system, which has a dynamic class hierarchy, message set, and method associations per class. The object system includes features such as multiple inheritance, property inheritance, and before and after methods [1]." Interleaf's "dynamic class hierarchy" shows the storing of graphically representable objects "based on membership in hierarchically organized classes.""

Claim 79 (dynamically linking a plurality of objects) is disclosed by Interleaf (p. 77): "Lisp was chosen as the I6 extension language because it is interpreted (providing a fast prototyping environment) and has run-time binding. Run-time binding is required for easy delivery of active documents." Interleaf's "run-time binding" shows the dynamic linking of a plurality of objects as recited in the claim.

Claims 80-82 (objects are managed by display list management module / display list management supports one page and multi-page documents at a plurality of levels / display list management can be expanded dynamically by new objects) is disclosed by Interleaf at p. 76 and "object-oriented structured document editor" at p. 83: "There must be programmatic access to do everything that is possible from the UI, and more. There should be access for object creation and destruction; object navigation (ideally through different views, such as format and structure); getting and setting predefined and user-defined properties .... There should be a set of editor objects, which can act as user agents to intercede on behalf of the user between the UI and the underlying document objects." These "editor objects" are display list management modules. Allowing a user to manipulate the document for "object creation and destruction," "object

navigation," and "getting and setting predefined and user-defined properties" shows the management of the objects by a display list management module as recited by the claims.

Interleaf teaches (p. 75) both the creation of simple documents as well as multi-page documents: "Interleaf... was originally designed for the creation of technical manuals that could be hundreds of thousands of pages in length. Such complex documents influenced the software's requirements, not just in performance and quantity handling, but also in its ability to address several organization and production workflow issues regarding documentation."

Using Interleaf to create and manage "complex documents" that "could be hundreds of thousands of pages in length" shows the support of one page and multi-page documents as recited in the claim. Interleaf teaches (p. 77) the use of the display list management module to dynamically expand the document with new objects: "This allows a user to simply cut and paste a document from an active system to a static one, then allowing the active document to introduce new classes and methods into the previously static environment." This shows that the display list management can be expanded dynamically by new objects (new classes and methods) as recited in the claim.

Ground #8: Claims 15, 19, 36, 52 and 58 are rejected under 35 USC 103(a) as obvious over Interleaf in view of the Interleaf patent, further in view of Lieberman. The Request at pp. 13-19 and Exhibit K claim chart are incorporated by reference. See Requester Comments 09/06/2011, p. 22, regarding Ground #8, which are incorporated by reference.

Regarding **claims 15, 19, 36, 52 and 58**, the combination of Interleaf/Interleaf Patent describes claim limitations as noted above. Interleaf/Interleaf Patent fail to disclose Java Script objects used as the formal language for the scripts. However, Lieberman evidences (p. 17) Java Script as a known scripting language. The limitations are met by substituting Java script objects read out to the interface for editing, transforming automatically into Java Script objects as described by Interleaf Patent 1: 15-27.

Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to combine the teachings of Interleaf, Interleaf Patent, and Lieberman because 1) The specific language used in scripting is "merely a matter of obvious engineering choice," and thus prima facie obvious. (MPEP § 2141.04.V.(B)). The '789 patent suggests no unexpected or unpredicted benefit of using Java Script in this manner, and in fact merely refers to Java Script as one of several "example" scripting languages that could be used. ('789 patent, 6:10-11). Thus, Lieberman teaches that substituting Lisp (as taught in Interleaf) for Java script is just substituting one known solution (Lisp) for another known solution (Java script). 2) A person of ordinary skill in the art would be motivated to combine Interleaf with the Interleaf Patent and Lieberman because they both address the same field, namely scripting interfaces for object systems. This overlap in the field of use would motivate a person of ordinary skill in the art to combine Interleaf, the Interleaf Patent, and Lieberman.

Ground #9: Claims 1-53, 54, 56-60, 62, 64-66, 68-71, 76-77, and 80-82 are rejected under 35 USC 103(a) as obvious over the Interleaf Patent in view of IBM. The Request (07/16/2010) at pp. 7-8 and 16-17 and Exhibit L claim chart are incorporated by reference. See Requester

Comments 09/06/2011, p. 22-23, regarding Ground #9 (and Ground #12 merged into Ground #9), which are incorporated by reference.

Regarding **claims 1-53**, The Interleaf Patent describes an "electronic document processing system" or EDPS that modifies documents for printing. Interleaf Patent at 1:8-11; 2:10-11. Fig. 5 of the patent, reproduced below, shows an example document in the form of a memorandum 32 having multiple objects, including a "graphic object 48." Interleaf Patent, 3:4. The Interleaf Patent describes providing scripts to the objects to trigger events during presentation or printing. A list of example events is provided in columns 6-9 of the patent. Interleaf Patent creates electronic documents for "printing." In reference to Interleaf Patent, when printing, the objects of the active document to be printed are combined into an output stream or alternatively, combining the objects of a document to be printed into an output stream would be obvious.

Generating such documents from an input print data stream was well known in the art, as shown by IBM. IBM more explicitly discloses (p. 8, printed from the AS/400; p. 270) printing a document defined in a high level print description language and Postscript (p. 8), a page description language. For example, IBM refers to several different types of output streams that include combined objects, including IPDS (Intelligent Printer Data Stream), HP's PCL (Printer Command Language) data stream, and Lexmark's PPDS (Personal Printer Data Stream). (IBM, pp. 15-16, 270) IBM describes the Advanced Function Presentation and printing system (AFP). As noted above IBM teaches (See Request, 07/16/2010, p. 7 & limitations addressed in Ground

#1 above) all the claim elements that were found to be missing in the original prosecution. IBM discloses teachings as noted in the first SNQ above that map to limitations of all claims 1-53, except for the Java Script limitations of claims 15, 19, 36, and 52. Java Script is an obvious, well known formal language for scripting. The specific language used in scripting is "merely a matter of obvious engineering choice," and thus prima facie obvious. (MPEP § 2141.04.V.(B)). The '789 patent suggests no unexpected or unpredicted benefit of using Java Script in this manner, and in fact merely refers to Java Script as one of several "example" scripting languages that could be used. ('789 patent, 6:10-11).

Reasons to Combine the Interleaf Patent and IBM

Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention of the '789 patent to combine the teachings of Interleaf Patent and IBM for the following reasons: 1) A person of ordinary skill in the art would have reason to combine IBM and the Interleaf Patent because they address the same field, namely object-oriented document publishing systems with scripting functionality. The above-described combination of prior art elements is done according to known methods to yield predictable results. See MPEP 2141 .III.(A). 2) Combining the printing process of IBM that parses and transforms a print data stream into an object-oriented format, and the print-scripting process of the Interleaf patent are combining well known printing techniques to yield predictable results. See MPEP 2141.III.(C), (D). 3) The Interleaf Patent suggests the combination. The Interleaf Patent describes an electronic document processing system ("EDPS") that operates on a computer workstation. (Interleaf Patent 2:10-12). "The workstation is connected to an Ethernet network, thereby allowing the workstation to access documents and software stored on other nodes 11." (Interleaf

Patent 2:15-16). The EDPS processes these documents for "creating, editing, printing and presenting." (Interleaf Patent 1:9-11). IBM describes a system that could be connected to the Interleaf patent workstation via the Ethernet network. (IBM, p. 262.) IBM describes a system that parses an input data stream to form documents. (IBM, p. 194). The documents produced by IBM are the same type of documents that the Interleaf Patent's EDPS processes and prints, as shown above. Therefore, one of skill in the art would be motivated by the teachings from the Interleaf Patent to combine these references. *See* MPEP 211.III.(G),

Claim 54 recites limitations similar to claims 1 and 17 and is rejected for the same reasons.

Regarding **Claim 56,** Interleaf Patent states (1:8-11): "This invention relates to a Document Processing System for creating, editing, printing and presenting active electronic documents which are associated with computer programs."

IBM more explicitly addresses the printing aspects (p. 1): "Print with complete system management and full error recovery, whether the printer is system-attached via twinax or network-attached via TCP/IP." Also, to the extent that claim 56 may refer to the printing of a document defined in a high-level print description language on a low-level bitmap-compatible printer, IBM teaches: "HPT converts the AFP data stream into HP PCL or Lexmark PPDS print data streams. This transform works in one of two ways: raster or mapping. Raster mode builds an image of the page in AS/400 memory to send to the printer ...."(IBM at 270) Building the "image of the page" in raster mode shows the printing of a high-level print data stream (PCL or PPDS) into a low-level bitmap image (the "image of the page" in raster mode) for printing on a less-

capable printer that does not have a PCL or PPDS printer driver specifically installed. See motivation to combine Interleaf Patent and IBM above.

Claim 57 recites limitations similar to claims 1 and 18 and is rejected for the same reasons.

Claim 58 recites limitations similar to claims 1, 16, and 19 and is rejected for the same reasons.

Claim 59 recites limitations similar to claims 1, 17 and 20 and is rejected for the same reasons. See discussion above regarding "analyze syntax."

Claim 60 recites limitations similar to claims 1 and 13 and is rejected for the same reasons.

Claim 62 recites limitations similar to claim 56 and is rejected for the same reasons.

Claim 64 recites limitations similar to claims 1, 17, and 18 and is rejected for the same reasons. See discussion above regarding "analyze syntax."

Claim 65 recites limitations similar to claims 1 and 18 and is rejected for the same reasons.

Claim 66 recites limitations similar to claim 56 and is rejected for the same reasons.

Claim 68 recites limitations similar to claims 1, 17, and 21 and is rejected for the same reasons. See discussion above regarding "analyze syntax."

Claim 69 recites limitations similar to claims 1, 13, and 60 and is rejected for the same reasons.

Claim 70 recites limitations similar to claims 1, 5, and 61 and is rejected for the same reasons.

Claim 71 recites limitations similar to claim 64 and is rejected for the same reasons.

Claim 76 recites limitations similar to claim 1 and is rejected for the same reasons.

Claim 77 recites limitations related to syntax analyzer, similar to claims 54, 59, 64, 68, and 72 and is rejected the same reasons. See discussion above regarding "analyze syntax."

Claims 80-82 recites limitations similar to claim 1 and is rejected for the same reasons. Interleaf Patent teaches (1:30-39) an improved document processing system for creating computer procedures (i.e., methods) and associating them with an electronic document structured as a nested hierarchy of constituent objects. Interleaf Patent is silent regarding "display list management."

IBM more explicitly teaches features related to display list management. See rejections of claims 80-82 under Ground #1 (anticipated by IBM) above. The motivation to combine the references is noted above.

Ground #10: Claim 55 is rejected under 35 USC 103(a) as obvious over Interleaf in view of the Interleaf Patent and USPN 6,678,705. See Requester Comments 09/06/2011, pp. 23, 24, & 33-35 which are incorporated by reference.

Claim 55 is dependent on claim 54, which is rejected above as obvious over the combination of Interleaf in view of Interleaf Patent (Ground #7). Claim 55 is very similar to issued claims 5 and 8 (as rejected above under the combination of Interleaf in view of Interleaf Patent), except that it is limited specifically to sending and receiving e-mails. Sending and receiving emails in response to scripts was well known in the art. For example, the Interleaf Patent teaches a "WYSIWYG document that can send itself over an electronic mail system." (Interleaf Patent at 7:50-51; *see also,* 8:19-22.). Interleaf and Interleaf Patent fail to expressly disclose "the data processing unit is further programmed to send and receive e-mails in the cases defined in the script."

More explicitly, '705 teaches (Ex. S at 3:17-22, 40-50) email messages via a script: "[T]he user sends the document to be archived as an attachment to an email addressed to the public folder that should contain the document, e.g. abc@saveme.com, where abc is the name of a public folder on the archiving server saveme.com 120 .... the deposit of a message in a folder triggers the Folder::OnMessageCreated event. As a result of the triggered event, a server-side script associated with this event is invoked and is passed the folder identification and the message identifier of the newly posted message. The script at step 122 first collects the properties of the message (date, sender, etc.) that can be automatically derived and assigns a unique id to the document. Then, it parses the body of the message and, if the body of the message is a form, it extracts all information contained in the form." This shows that sending and receiving email messages via a script were both well-known functions in the prior art well before the priority date of the '789 patent.

A person of ordinary skill in the art at a time before the invention of the '789 patent would have combined Interleaf, the Interleaf Patent, and the '705 Patent for the following reasons:

Interleaf, the Interleaf Patent, and the '705 patent all address a common subset of functionality, that is, the programmatic parsing and use of input streams by scripts. Combining the scripted printing functionality of Interleaf and the Interleaf patent with the message scripting of the '705 Patent are combining well known printing techniques to yield predictable results. See MPEP 2141.III.(A). Interleaf suggests the combination because it is explicitly designed to leverage outside printing and publishing functionality: "External publishing functionality. In addition to having the presentation and editing functionality within the document, document system based applications can also take advantage of all the other publishing facilities that exist within the document system. For example, the IMA application makes use of the system's printing and filtering capabilities." (Interleaf, p.82)" The external email functionality of the '705 patent is one example of such external publishing facilities. Interleaf and the Interleaf Patent are directed to the same physical system. The existence of this system serves as an explicit suggestion to combine the teachings of Interleaf and the Interleaf Patent both of which describe the system. This would motivate a person of ordinary skill in the art to combine Interleaf and the Interleaf Patent to gain more information about the underlying product. A person investigating the '705 patent would be motivated by "common sense" to find other descriptions of known scripting functionality. This "common sense" would motivate a person of ordinary skill in the art to combine the '705 Patent with Interleaf and the Interleaf Patent, which have similar scripted functionality.

Ground #11: Claim 55 is rejected under 35 USC 103(a) as obvious over IBM in view of Interleaf, Interleaf Patent and USPN 6,678,705. See Requester Comments 09/06/2011, pp. 23, 24, & 33-35 which are incorporated by reference.

Claim 55 is dependent on claim 54 (as rejected under Ground #3, IBM, Interleaf, Interleaf Patent). Claim 55 is very similar to issued claims 5 and 8 (as rejected under Ground #3), except that it is limited specifically to sending and receiving e-mails. Sending and receiving emails in response to scripts was well known in the art. For example, the Interleaf Patent teaches a "WYSIWYG document that can send itself over an electronic mail system." (Interleaf Patent at 7:50-51; *see also,* 8:19-22.)]. IBM, Interleaf, Interleaf Patent fail to expressly teach sending and receiving email messages in the cases defined in the script.

More explicitly ‘705, teaches email messages via a script. See ‘705 teachings in Ground 10 above.

A person of ordinary skill in the art at a time before the invention of the '789 patent would have combined IBM, Interleaf, the Interleaf Patent, and the '705 Patent for the following reasons: A person of ordinary skill in the art would have reason to combine IBM, Interleaf and the Interleaf Patent because they address the same field, namely object-oriented document publishing systems with scripting functionality. A person of ordinary skill in the art would be further motivated to combine the above references with the '705 patent because the '705 patent shows well-known scripting functionality that was available for all scripted systems in the prior art. The above-described combination of prior art elements is done according to

known methods to yield predictable results. See MPEP 2141 .III.(A). The IBM, Interleaf, Interleaf Patent, and '705 patent all address a common subset of functionality, that is, the programmatic parsing and use of input streams by scripts. Combining the printing process of IBM that parses and transforms a print data stream into an object-oriented format, and the print-scripting process of Interleaf, the Interleaf patent, and the message scripting of the '705 Patent are combining well known printing techniques to yield predictable results. See MPEP 2141 .III.(C), (D). Interleaf suggests the combination because it is explicitly designed to leverage outside printing and publishing functionality: "External publishing functionality. In addition to having the presentation and editing functionality within the document, document system based applications can also take advantage of all the other publishing facilities that exist within the document system. For example, the IMA application makes use of the system's printing and filtering capabilities." (Interleaf, p.82)" The email functionality described in the '705 patent is an example of one outside publishing facility. IBM provides external printing and publishing functionality: "Advanced Function

Presentation (AFP) is a printing and presentation system that enables you to take full advantage of printing technology on your AS/400 system." IBM also teaches the desire to control outside printing and publishing functionality. "[P]ostprocessing option. A hardware device that attaches to the output side of a printer; for example, an envelope stuffer, binder, or stapler." (IBM, p. 404) A person of ordinary skill in the art would be motivated to "take full advantage" of the printing and publishing capabilities of the IBM system by combining it with the Interleaf system. See MPEP 2141.III.(G) Interleaf and the Interleaf Patent are all directed to the same physical system. The existence of this system serves as an explicit

suggestion to combine the teachings of Interleaf and the Interleaf Patent, all of which describe the system. This would motivate a person of ordinary skill in the art to combine Interleaf and the Interleaf Patent to gain more information about the underlying product. A person investigating the '705 patent would be motivated by "common sense" to find other descriptions of known scripting functionality. This "common sense" would motivate a person of ordinary skill in the art to combine IBM, the '705 Patent, Interleaf, and the Interleaf Patent.

Ground #12 is consolidated into original Ground #9 as both propose rejections obvious over Interleaf Patent in view of IBM.

Ground #13: Claims 55, 63, and 67 are rejected under 35 U.S.C. 103(a) as being obvious over the Interleaf Patent in view of IBM and the '705 patent. See Requester Comments 09/06/2011, p. 23-28 and 36-37, which are incorporated by reference.

Claims 55, 63, 67 (dependent on claim 54, 59, 64 respectively, rejected under Ground #9, obvious rejection Interleaf Patent in view of IBM) recite limitations similar to claims 5 and 8 (as rejected under Ground #9) and are rejected for the same reasons. Claim 55, 63, and 67 specifically limit to sending and receiving e-mails. Sending and receiving emails in response to scripts was well known in the art. For example, the Interleaf Patent teaches a "WYSIWYG document that can send itself over an electronic mail system." (Interleaf Patent at 7:50-51; *see also,* 8:19-22.)]. Interleaf Patent and IBM fail to expressly teach sending and receiving email messages in the cases defined in the script.

More explicitly '705, teaches email messages via a script. See '705 teachings in Ground #10 above.

A person of ordinary skill in the art at a time before the invention of the '789 patent would have combined Interleaf Patent, IBM and the '705 Patent for the following reasons: A person of ordinary skill in the art would have reason to combine IBM, and the Interleaf Patent because they address the same field, namely object-oriented document publishing systems with scripting functionality. A person of ordinary skill in the art would be further motivated to combine the above references with the '705 patent because the '705 patent shows well-known scripting functionality that was available for all scripted systems in the prior art. The above-described combination of prior art elements is done according to known methods to yield predictable results. See MPEP 2141 .III.(A). The IBM, Interleaf Patent, and '705 patent all address a common subset of functionality, that is, the programmatic parsing and use of input streams by scripts. Combining the printing process of IBM that parses and transforms a print data stream into an object-oriented format, and the print-scripting process of Interleaf patent, and the message scripting of the '705 Patent are combining well known printing techniques to yield predictable results. See MPEP 2141 .III.(C), (D). The email functionality described in the '705 patent is an example of one outside publishing facility. IBM provides external printing and publishing functionality: "Advanced Function Presentation (AFP) is a printing and presentation system that enables you to take full
advantage of printing technology on your AS/400 system." IBM also teaches the desire to control outside printing and publishing functionality. "[P]ostprocessing option. A hardware device that attaches to the output side of a printer; for example, an envelope stuffer, binder,

or stapler." (IBM, p. 404) A person of ordinary skill in the art would be motivated to "take full advantage" of the printing and publishing capabilities of the IBM system by combining it with the Interleaf system. See MPEP 2141.III.(G). A person investigating the '705 patent would be motivated by "common sense" to find other descriptions of known scripting functionality. This "common sense" would motivate a person of ordinary skill in the art to combine IBM, the '705 Patent, and the Interleaf Patent.

For the reasons above, claims 1-82 of USPN 6,684,789 B2 to Krautter have been reexamined. No claims have been confirmed.

Response to Arguments

The following response to Patent Owner's arguments incorporates the assertions, evidence and opinions presented in the Widuch, Picht, and Birnbaum declarations. The response to Third Party comments, likewise, incorporates the assertions, evidence and opinions presented in the Paul Jacobs declaration.

Secondary Considerations of Non-Obviousness and Commercial Success

Patent Owner (Patent Owner Remarks, 03/30/2011, pp. 1-4) notes the state of litigation in 2:09cv 04254. Patent Owner asserts that JScribe® is covered by the patent in this reexamination. **Patent Owner** cites to '789, 5: 55-59: "At this point, it should be mentioned that these systems operating by the method according to the invention are also designated JScribe (registered trademark) systems and, accordingly, the method according to the invention is also

designated JScribe (registered trademark)" and the Picht Dec ¶11-20. **Patent Owner** also cites to the Widuch Declaration ¶¶ 5-6 and asserts (p. 2) "IBM's and Samsung's extensive, worldwide exploitation of the technology at issue – and IBM's substantial payments for the right to license the technology - together seriously undermine Samsung's claim here that the technology was allegedly old, and had been used by IBM before the filing date. Obviously, if IBM already had the technology, as Samsung now claims, IBM would never have paid to license it."

Patent Owner summarizes (pp. 2-3) the IBM licensing of JScribe® technology, referencing the Widuch declaration ¶¶7 & 9 and Widuch Exhibit A. **Patent Owner** opines that the image of Exhibit A includes the statement: "This program is protected by copyright laws and U.S. patent 6,684,789." From that Patent Owner alleges an admission by IBM that JScribe® software is covered by the patent at issue and also shows evidence that IBM regarded the '789 patent as valid.

Patent Owner summarizes (pp. 3-4) the Samsung sub-license and the IBM-Samsung relationships (citing to the Widuch Declaration ¶¶ 10, 13, & 19, the Widuch Exhibits B, C, E, & G, and the Picht declaration ¶¶ 11-21). **Patent Owner** asserts that millions of dollars were paid in royalties, that Samsung and IBM have at various times claimed the JScribe® technology as their own, and notes product endorsement (Exhibit E, Samsung's press release of the advantages of CCP's product) and awards (Exhibit C, Best Seller Award).

Patent Owner notes (p. 3) the termination of the IBM License and asserts the loss of any rights that Samsung may have had under that agreement. **Patent Owner** references the Widuch declaration ¶13, Widuch Exhibit G, and the Picht declaration ¶¶11-20 for support. **Patent Owner** opines (pp. 3-4), "We discuss these indisputable facts in more detail below, but they are exceptionally strong proof of objective evidence that the '789 patent claims, which cover CCP's JScribe® product (Picht Dec. ¶¶11-21), would not have been obvious to those of ordinary skill. Had they been obvious, IBM and Samsung would [not] have paid millions of dollars to license the patented technology. (Examiner inserted "not", typo presumed.)

Patent Owner additionally argues (pp. 20-24) secondary considerations of non-obviousness and commercial success. **Patent Owner** cites to the Widuch declaration (¶¶7-8, 10) noting the millions of dollars paid in licensing JScribe® technology, as support for arguments related to commercial success. **Patent Owner** (pp. 21-23 & Widuch Exhibits B-G) cites to praise and continued promotion of the JScribe® product as support for the validity of the '789 patent. **Patent Owner** cites (p. 21) to *In Gambro Lundia AB v. Baxter Healthcare Corp.,* 110 F.3d 1573, 1579 (Fed. Cir. 1997) and *Accord Eli Lilly & Co. v. Zenith Goldline Pharmaceuticals, Inc.,* 471 F.3d 1369 (Fed. Cir. 2006) (pointing to the record showing a number of awards as indicators of industry acclaim). See also, *In re Tiffin,* 443 F.2d 394, 400 (CCPA 1971) (Recognition of advantages of [the patented feature] as relevant to a determination of nonobviousness. Recognition by the trade is the best and most persuasive evidence that can be offered.) **Patent Owner** asserts (pp. 23-24), "Despite all this contrary evidence, Samsung now says that IBM's AS/400 software is

identical to the '789 patent (i.e., IBM anticipated the claimed subject matter) and rendered obvious

the innovations claimed in the '789 patent. But the truth of the matter is that in 2004, four years *after* the patent's priority date, and five or more years after publication of the IBM Reference, IBM

paid handsomely to license CCP's software covered by the patent at issue. (Widuch Dec. ¶ 7). As a matter of common sense, if the innovations of the '789 Patent were so obvious (as Samsung now claims), and IBM already had identical or similar software, neither Samsung nor IBM would have paid anything to license CCP's software."

Mr. Picht opines (Picht Declaration, in support of Patent Owner, 03/30/3011, ¶11) that "Claim 1 of the '789 patent covers the JScribe® software when used for printing." The **Picht Declaration** (¶¶12-20) opines that to the extent that Samsung's printers do not include scripts assigned to objects, Samsung's printers with the JScribe® software installed have an interface that allows users to easily create scripts and to include them in the software. There is "[a]t least one Samsung US customer (identified in Widuch Dec. Ex. E, p, 5), which according to Samsung, has "[o]ver 1000" printers, where the printers also have scripts assigned to an object, including ones that are used for security equipment (see e.g., claims 4, 25, and 41). Similarly, I believe that CCP's JScribe® software, which I understand has been incorporated into Samsung printers, has the functionality to include each of the claim elements of claims 2-16 of the '789 patent. That is, users can create scripts that interface with the JScribe® software..." "In addition, as explained above, systems, like various Samsung printers which include the JScribe® software, that are

connected to a computer network are covered by claims 17-19. "For similar reasons, I believe that Samsung's printers which include JScribe® software are covered by claim 20. "...at least one Samsung US customer has a print server that includes JScribe® software. There are at least half a dozen other companies that have JScribe® software loaded on their print servers. All have scripts assigned to objects that perform various functions, including, for example, having timers and the ability to reassign data. (See e.g., claims 7, 13 and 21). Samsung printers, which include the JScribe® software, also are covered by claims 22-37 because they have a memory (i.e., a computer readable medium) on which the infringing method steps are stored. Scripts that users install on the printer would also be stored in memory, and these method steps would be carried out on the printer processor when the execution command is triggered (e.g., circumstances defined in the script, for example, a timer).

The **Widuch Declaration** (01/10/2011, ¶¶ 5-6, in support of Patent Owner) notes a number of Samsung printers were sold that include the JScribe® technology. **Widuch** opines that Samsung allowed download of JScribe® technology without permission. **Widuch** asserts (¶¶ 7-13) that IBM licensed, and was given the right to sub-license (to Konica-Minolta Business Solutions, Inc. and Samsung Electronics Co. Ltd.) under some certain conditions the JScribe® software. **Widuch** notes that (¶19) in 2009, CCP terminated the IBM License and with it any rights that Samsung may have had under that agreement. The **Widuch** declaration includes exhibits (all received 01/10/2011) that acknowledge licensing agreements and benefits / awards attributed to the JScribe® technology:

Exhibit A – JScribe® Print Server Solution 1.0 IBM, build 1.02, undated; Widuch Dec ¶9, "IBM and CPP co-marketing logo license agreement is attached as Exhibit A..." (alleged as attached to license agreement indicating acceptable marking effort asserting nexus of patent to jscribe)

Exhibit B – Alliance in Printing, Samsung and IBM Korea, undated; Widuch Dec ¶11, IBM and Samsung joint press release. In support of Widuch opinion that IBM and Samsung took credit for the JScribe® technology.

Exhibit C – dated 01/26/2006, non-English document, @ p. 3, "BestSeller Award 2005: Kategorie Service: CCP Systems AG, ONVENTIS GmbH"; Widuch Dec ¶12 notes recognition by IBM of CPP's JScribe® technology.

Exhibit D - Chinict 2007 Top ICT Innovators Press Release "CCP Systems wins the European Commission ChinICT Innovator Award"; Widuch Dec ¶14 points out that in June 2007, CCP's JScribe® product won the European Commission China Information and Communications Technologies Innovator Award (ChinICT). CCP was selected from 60 enterprises as among the best innovators.

Exhibit E – "New Samsung MFP with advanced features offers easy integration for efficient, simple and reliable printing, January 2009; Widuch Dec ¶15 provides a press release and opines that Samsung characterized the advantages of Samsung's JScribe®.

Exhibit F – JScribe Samsung White Paper, undated; Widuch Dec ¶16 cites to a white paper noting that in about 2008-09, Samsung praised the patented JScribe® technology. Widuch Dec ¶17 opines that "all the deals cited in the Samsung White Paper could only

have been done because of the JScribe® technology in the Samsung devices. Widuch Dec ¶18 quotes from the White Paper, p. 1: "JScribe is an embedded application and communication platform for MFPs [multifunctional printers]. The JScribe® software development Kit (SDK) allows third-parties the flexibility to shape and extend the functionality of Samsung's MFPs to meet their specific business requirements."

Exhibit G – "Our thoughts on designing printers and printing solutions", dated 12/15/2000; URL<http://www.samsung.com/us/b2b/learning/whitePapers.html> (Single page screen shot.); Widuch Dec ¶20 notes that [e]ven after having been sued, and after bringing on this reexamination, Samsung's web site continues to promote the advantages of "JScribe® technology"; "As printers have gone from only printing to complicated multi-function machines, Samsung has provided JScribe® software to support all the new functions and features."

Third Party Requester (TPR 09/06/2011, p. 2) opines that Patent Owner remarks related to litigation are erroneous and/or irrelevant assertions. **Requester** asserts that the licenses that occurred in 2006 were software licenses that provided source code to the licensees. **Requester** makes note (pp. 14-15) that secondary considerations are to be considered as evidence to rebut obviousness rejections under 35 U.S.C. § 103 and has no bearing on the anticipation rejections of claims 1-7, 11-12, 17, 20-28, 32-33, 38-44, and 48-49. Requester states the following: "[t]o be of probative value, any secondary evidence must be related to the claimed invention (nexus required)." *Id.* The MPEP continues to say "[t]he term 'nexus' designates a factually and legally sufficient connection between the objective evidence of nonobviousness and the claimed

invention so that the evidence is of probative value in the determination of nonobviousness." *Id.*, citing *Demaco Corp. v. F. Von Langsdorff Licensing Ltd.*, 851 F.2d 1387 (Fed. Cir. 1988). The PO argues that their "JScribe® Technology" has been widely licensed and praised in the industry, which should be a consideration to support a finding of non-obviousness. (PO Remarks at 21-23.) The licenses for JScribe® were all software licenses, where actual source code was provided. The PO has not asserted that the '789 patent was even mentioned in the licenses. Nor has the PO shown that the licensees entered into the licenses in order to obtain the allegedly patentable aspects of the source code. In addition, the prior existence of three software licenses -- which are now terminated -- does not show that the technology is widely licensed in the industry. The PO does not point to any other use of JScribe® in the industry, including any company that is using it today. Third, MPEP 716.01 (d) requires that any objective evidence of non-obviousness be weighed against the evidence supporting a prima facie case of obviousness. "Although the record may establish evidence of secondary considerations which are indicia of nonobviousness, the record may also establish such a strong case of obviousness that the objective evidence of nonobviousness is not sufficient to outweigh the evidence of obviousness." Id. The prior art in the present request is extremely strong, and actually anticipates 30 of the 53 claims, including all of the independent claims. The PO has not shown how any of the alleged evidence of commercial success is directed to the elements of the dependent claims that stand rejected as being obvious. Therefore, the record in the present case has established a strong case of obviousness that any evidence of commercial success of the JScribe® product is clearly insufficient to overcome the obviousness conclusion.

In reference to Mr. Picht's comment (Picht Dec 03/30/2011, ¶19), **Examiner** notes that a "data processing unit" is identified in the specification, but not a "printer processor." See teaching at '789, 8: 2-8: "...system... comprising at least one data processing unit..." or 8: 57-61: "From a computer 1, an input print data stream 2 is sent to a device 3-for example a computer such as a PC or else an intelligent output device such as an intelligent printer – which operates in accordance with the method according to the present invention." An "intelligent printer" is not defined in the specification. It is unclear what is meant by a "printer processor." Picht also recites a phrase (¶19): "scripts that users install on the printer." There is no explicit teaching of installing a script on a printer in the specification.

To address the probative value of expert opinion, **Examiner** has considered the nature of the matter sought to be established, the strength of any opposing evidence, interest of the expert in the outcome of the case, and the presence or absence of factual support for the expert's opinion. *Ashland Oil, Inc. v. Delta Resins & Refractories, Inc.*, 776 F.2d 281, 227 USPQ 657 (Fed. Cir. 1985), cert. denied, 475 U.S. 1017 (1986). See also *In re Oelrich*, 579 F.2d 86, 198 USPQ 210 (CCPA 1978) (factually based expert opinions on the level of ordinary skill in the art were sufficient to rebut the prima facie case of obviousness); *Ex parte Gray*, 10 USPQ2d 1922 (Bd. Pat. App. & Inter. 1989) (statement in publication dismissing the "preliminary identification of a human b-NGF-like molecule" in the prior art, even if considered to be an expert opinion, was inadequate to overcome the rejection based on that prior art because there was no factual evidence supporting the statement); *In re Carroll*, 601 F.2d 1184, 202 USPQ 571 (CCPA 1979) (expert opinion on what the prior art taught, supported by documentary evidence and formulated

prior to the making of the claimed invention, received considerable deference); *In re Beattie*, 974 F.2d 1309, 24 USPQ2d 1040 (Fed. Cir. 1992) (declarations of seven persons skilled in the art offering opinion evidence praising the merits of the claimed invention were found to have little value because of a lack of factual support); *Ex parte George*, 21 USPQ2d 1058 (Bd. Pat. App. & Inter. 1991) (conclusory statements that results were "unexpected," unsupported by objective factual evidence, were considered but were not found to be of substantial evidentiary value).

Regarding the Patent Owner arguments (and related Widuch, and Picht declarations and exhibits) alleging evidence of non-obviousness, **Examiner** confirms Requester's point that such evidence of secondary considerations is irrelevant to rejections under 35 U.S.C. 102 and thus cannot overcome a rejection so based. *In re Wiggins,* 488 F.2d 538, 543, 179 USPQ 421,425 (CCPA 1973). Accordingly, the declarations will be addressed to the extent that they (1) provide evidence of commercial success, in general, and (2) are directed specifically to claims rejected under 35 USC 103(a).

Examiner finds that the Declaration of Widuch, in support of Patent Owner, ¶¶1-4 presents credentials of one skilled in the art. ¶¶5-20 present opinions and a timeline of events involving Samsung, IBM and JScribe software related to licensing, awards, and press releases. Widuch provides an expert opinion (¶17): "In my opinion, all the deals cited in the Samsung White

Paper (¶15, Exhibit F) could only have been done because of the JScribe technology in the Samsung devices.

Examiner finds that the Declaration of Picht, in support of Patent Owner, ¶¶1-6 presents credentials of one skilled in the art. As the Director of Business Development at CPP Systems AG, it is presumed that Mr. Picht has a vested interest in the outcome of the reexamination. Mr. Picht asserts that he has read and understands the '789 patent and the functionality of CCP's JScribe product. Mr. Picht describes the licensing and sub-licensing arrangements. Mr. Picht describes JScribe as incorporated into many of the Samsung printers and multifunctional devices using elements of the claims of '789. Mr. Picht states (¶20), "I have reviewed the proposed new claims that CCP has submitted in this reexamination, and believe the new claims CCP has presented here cover CCP's JScribe® technology as discussed above."

New evidence provided must be relevant to the issues raised in the rejection. For example, declarations in which conclusions are set forth without establishing a nexus between those conclusions and the supporting evidence, or which merely express opinions, may be of limited probative value with regard to rebutting a prima facie case. *In re Grunwell*, 609 F.2d 486, 203 USPQ 1055 (CCPA 1979); *In re Buchner*, 929 F.2d 660, 18 USPQ2d 1331 (Fed. Cir. 1991). See MPEP § 716.01(a) through § 716.01(c). To be of probative value, any secondary evidence must be related to the claimed invention (nexus required). The term 'nexus' designates a factually and legally sufficient connection between the objective evidence of nonobviousness

and the claimed invention so that the evidence is of probative value in the determination of nonobviousness (citing Demaco Corp. v. F. Von Langsdorff Licensing Ltd., 851 F.2d 1387 (Fed. Cir. 1988)). Reexamination does not address alleged violations of license agreements. **Examiner** cannot confirm that any alleged license agreements did or did not include USPN 6,684,789 B2 to Krauter. **Examiner** finds no supporting evidence provided regarding Requester's comment (09/06/2011, p. 2) that Patent Owner remarks related to litigation are erroneous and/or irrelevant assertions. **Examiner** notes that no copies of licenses have been evidenced. **Examiner** cannot confirm the unsupported assertion by Patent Owner that Samsung sold many printer units using JScribe® technology. Additionally **Examiner** notes that any asserted use of JScribe® technology within devices provides limited evidence of commercial success because these devices that allegedly use JScribe® technology are very complex and there may be numerous features that account for any (alleged or actual) success in the market. The weight to be given any objective evidence is made on a case-by-case basis. The mere fact that an applicant has presented evidence does not mean that the evidence is dispositive of the issue of obviousness. The question of obviousness must be resolved on the basis of these factual determinations. While each case is different and must be decided on its own facts, the Graham factors, including secondary considerations when present, are the controlling inquiries in any obviousness analysis.

Patent Owner has provided (pp. 4-6) summaries of the '789 patent and the prior art references applied in rejections. **Patent Owner** asserts (p. 4), "The '789 patent claims cover CCP's highly successful software product called JScribe®." (Cites to Picht Dec. ¶¶11-20; see also Widuch

Dec. ¶¶7-18 and 20.) Patent Owner cites (p. 4) to '789, 8: 55-65 for support of the software functions asserting, "The software, which typically resides in printers or multi-function devices, receives an input print stream and parses the print steam into objects, which are then stored in object-oriented format...At least one script is assigned to an object, which is executed in the cases defined by the script. When sent for priming, the objects are transformed into a format to control a printer, and are combined into an output print data stream." Patent Owner provides a figure (p. 4) to support their statement.

Patent Owner summarizes (pp. 4-5 & cites to Birnbaum Dec. ¶8) the IBM AS/400 computer system. **Patent Owner** opines (p. 5), "Requester has purported to find all claim elements in the IBM Reference, but the similarities are superficial, at best. Closer inspection of the IBM Reference reveals that essential components of claims are entirely missing, for example: • parsing the input data stream; • storing graphically representable objects in objected-oriented format; • using scripts; and • assigning at least one script to an object."

Patent Owner provides an opinion of the (p. 5) Interleaf Active Documents ("Interleaf"): "The Interleaf Documents describe so-called "active documents." However, "active documents" are created on a computer, far upstream from the data discussed in the IBM reference (or the '789 patent) (see Figure in Birnbaum Dec ¶9). "Active documents" may be printed; however, the only print stream is the output of a print process, and print stream treatment is not discussed in any detail in the Interleaf documents. (Birnbaum Dec. ¶9) The technology in the '789 patent, by

contrast, covers methodology that *prints* documents, *after* they have been created by other programs; that is, the print data stream is an input, which is then transformed through the patented technique into an output data stream. Transforming a print data stream is not disclosed or suggested in the Interleaf Documents."

Patent Owner provides an opinion of the Lieberman reference: "Lieberman describes automating user interaction by using scripts. (See Figure in Birnbaum Dec. ¶ 10). Lieberman is really far afield from the patented technology. Lieberman is not related to printing at all, and cannot be considered the same field as the '789 patent. Therefore, combining these disparate references is not something a person of ordinary skill would have done."

Patent Owner provides (pp. 5-6) a drawing relating the prior art and the '789 patent within a document creation and printing "food chain." (Birnbaum Dec ¶ 11) **Patent Owner** opines (p. 6), "...the IBM Reference, Interleaf Documents, and Lieberman are each at different stages of the print workflow, and do not contain the claim elements. Only with improper hindsight - and complete system redesigns - would it have been "obvious" to "cherry-pick" features of each of these references to create the inventions of the '789 patent."

The **Birnbaum Declaration** summarizes (¶¶8-11) IBM AS/400, Interleaf, Interleaf Patent, and Lieberman and provided a drawing representing the related location in the document creation

and printing "food chain." Regarding IBM, **Dr. Birnbaum** opines (Birnbaum Dec ¶8), "The AS/400 receives data from an application (including text, images, etc.), prepares the data (including, for example, by executing a DDS (Data Description Specification) on the input print data) as output print data, and sends this output data to the selected printer." "Regarding Interleaf, **Dr. Birnbaum** opines (Birnbaum Dec ¶9), "Active documents are created at the computer, far upstream from the data print stream discussed in IBM (or the '789 patent). Documents in the Interleaf references may certainly be printed; however, the only print stream is the output of a print process, and the treatment of this print stream is not discussed in detail in the Interleaf documents."

Dr. Birnbaum notes (Birnbaum Dec ¶10) that Lieberman describes "automating user interaction by using scripts."

Third Party Requester asserts (p. 2) that they do not agree with Patent Owner's "Factual Background of The Reexamination" nor with the "Introduction To The Technology" section. **Requester** does not agree with Patent Owner's Generated schematic diagram (PO Remarks, p. 6) that characterizes the references, but notes Patent Owner's agreement that all of these references are part of a printing process or "chain."

Examiner addresses any related art arguments in turn. **Examiner** does point out that the '789 patent includes a method and system as part of a printing process (see '789, FIG. 1) that inputs a data stream, stores, transforms, outputs to an output device, and receives feedback. **Examiner**

asserts that hindsight (See discussion below, ACP p. 140) is not applied when citing to the features of a print process.

Regarding Ground of Rejection No. 1: Claims 1-5, 17, 20-26, and 38-41 under 35 U.S.C. § 102(b) as anticipated by the IBM reference. See Request pp. 7-8, 10-12, and Exhibit G/Claim Chart for SNQ #1, 07/16/2010, anticipation based on IBM. **Patent Owner** argues (pp. 6-9) that IBM does not disclose a parser to analyze a print data stream. **Patent Owner** asserts (p. 6) a parser disclosed at '789, 3: 21-26 (In this case, as opposed to the use of a status machine, the analysis and splitting of the input print data stream by a parser (syntax analyzer) ensures that the syntax of the page description language is no longer restricted to the use of regular expressions, and thus powerful page description languages can also be used.) and at '789, 4: 32-38 (...parser is used for the analysis and splitting up into the graphically representable objects ... which is therefore capable of analyzing and splitting up languages with "context-free grammars"). At footnote 1 (p. 6), **Patent Owner** explains: "The term "status machine" used in the '789 patent is likely an incorrect translation from the original German and should be understood as a "state machine." " **Patent Owner** opines (pp. 6-7), "A parser as properly understood according to the '789 patent, therefore, is a syntax analyzer: it is capable of analyzing the syntax of an input data stream based on a formal grammar. (Birnbaum ¶12). "Parser" in the '789 patent is used consistently with its well-accepted meaning in computer science. "This functionality permits the parser to analyze complex page description languages (PDLs), in comparison to a state machine driven by regular expressions, which (as shown below with respect to the IBM Reference) can only reasonably handle single commands in a line by line approach. (Birnbaum ¶13). It should be

noted that PDLs do not have "input print lines" and therefore, the statement in the IBM reference about "parsing" is not applicable to PDL input data streams, including Postscript. There is absolutely no disclosure in IBM on whether (and, if so, how) PDL inputs are "parsed" within any reasonable understanding of the term as used in the patent claims. (Birnbaum ¶18). **Patent Owner** asserts, "Samsung purports to find a parser at IBM reference, pp. 194-195, including: "Specifically, the page definition defines how data is placed on a logical page layout. Input print lines are read in, optionally parsed into individual fields, and place [sic] on the page." "However, this function is capable of being performed by a state machine, and would not require a parser, as described and claimed in the '789 patent. (Birnbaum Dec. ¶14). This difference would have been understood by those of ordinary skill in computer science generally, and in the field of the invention in particular. Specifically, as discussed below, by referring to the data fields described at IBM Reference, pp. 14-15, it is clear that input print lines are composed of structured fields. Identifying and separating the individual fields from a structured field is a simpler operation than that required for parsing, and can be performed by a state machine. (Birnbaum Dec. ¶15)."

Patent Owner asserts (footnote 2, p. 7), "In the same document, IBM states that "the print line is then parsed (sub-defined into individual fields) to define the zip code field" (p. 215, emphasis added), further demonstrating the IBM Reference's loose use of the term "parsing" unrelated to syntactic analysis. "The input process described in the IBM Reference involves receiving lines in SCS (SNA [System Network Architecture] Character Stream) or IPDS (Intelligent Print Data Stream) format (defined below), and converting data fields in such lines into respective fields in

AFPDS (Advanced Function Print Data Stream). (Birnbaum ¶16)." Regarding SCS, the IBM Reference explains: "The SNA character string (SCS) data stream has a relatively simple structure, consisting of a one-byte hexadecimal code followed by the data to be printed. SCS, which is the standard pre-AFP print data stream, is used to control line printers and supports row and column functions." (IBM Reference, p. 14, emphasis added)

Patent Owner asserts (p. 8), "In particular, "[t]he structured field format of IPDS allows commands to be sent to the printer in a continuous stream. Each command is self-describing." (Birnbaum Dec. ¶17; and Exs. B at 492 / IBM @server iSeries Printer Device Programming, v 5, September 2002, and C at 555 / AS/400e Printer Device Programming v4, May 1999; all received 03/30/2011). That is, IPDS has no grammar and no syntax across commands. (Birnbaum ¶17) **Patent Owner** asserts (p. 8) that (IBM, pp. 14-15) input print lines are made up of structured data fields...This operation of identifying and separating the individual data fields in a structured format is a different and simpler operation than parsing. (Birnbaum ¶18).

Patent Owner asserts (p. 9), citing to Birnbaum ¶19: "A field in AFPDS is identified by a leading 0X5A byte, and is followed by a number of fields that identify the length of the data, the type of data "ID", a flag byte and a sequence number, followed by the data. (Birnbaum Dec. ¶20). This may be understood by reference to the figure in Birnbaum Dec ¶6. The critical distinction is this: **the location and content of the structured fields are predefined and do not require parsing, since their location is already defined, as is the meaning of the values in**

each field. (Birnbaum ¶21). Thus, evaluating and processing input print lines in the IBM Reference, which is defined as structured fields, can be done by a series of if-then statements. (Id.) Such an evaluation is one implementation of a state machine, which the '789 patent expressly distinguishes from the parser used in the invention. (See '789 patent, Col. 3, lines 21-30; Birnbaum ¶21). Indeed, IBM says: "IPDS printers are *state machines...*" (Birnbaum Ex. C at 557, received 03/30/2011; emphasis in original). Even though IBM AS/400 can handle Postscript documents as an input - it cannot parse them into graphically representable objects. This is because the only methodology disclosed in IBM requires conversion into structured fields. (See IBM p. 194: input print lines are "parsed into individual fields" (Birnbaum ¶22). In contrast, parsing, as used in the '789 patent, is a far more complex operation; it requires scanning the input data stream character by character, identifying tokens, e.g., collections of characters that correspond to commands or data and a syntax evaluation to determine the relationships between the identified tokens. (Birnbaum ¶22). Often there is no rigid format for the input data stream. (Id.) The IBM reference, in contrast, only discloses conversion from structured SCS or IPDS format to AFPDS. That function is not parsing under any reasonable interpretation of the patent claims, because it does not require syntax analysis. (Birnbaum ¶23). For this reason alone, the IBM reference did not anticipate the '789 patent claims.

The Birnbaum Declaration (¶¶12-23) and Exhibits B, C (all received 03/30/2011) provide an opinion and supporting evidence related to the term "parser," as outlined in Patent Owner arguments above. Exhibit B (IBM eserver iSeries Printer Device Programming, v 5, September 2002) at page 492 explains processing IPDS commands: "The structured field format of IPDS

allows commands to be sent to the printer in a continuous stream..." Similarly, Exhibit C (AS/400e Printer Device Programming v4, May 1999) at page 555 describes processing IPDS commands.

Dr. Birnbaum (¶¶8-11) provides a summary of the cited art and an opinion (¶¶12-13) on the term "parser": "A parser, as properly understood according to the '789 patent, is a syntax analyzer: it is capable of analyzing the syntax of an input data stream based on a formal grammar." "The "parser" in the '789 patent is consistent with a well-accepted meaning of the term in computer science. This functionality permits the parser to analyze even complex page description languages (PDLs), in comparison to a state machine driven by regular expressions, which can only reasonably handle single commands in a line by line approach." **Dr. Birnbaum** opines (¶¶14-23) on a parser as related to the IBM prior art: "The function described in IBM at pp. 194-195 is capable of being performed by a state machine, and would not require a parser..." "By referring to the data fields described at IBM, pp. 14-15, it is clear that input print lines are composed of structured fields. Identifying and separating the individual fields from a structured field is simpler than parsing, and can be performed by a state machine." "The input process described in IBM involves receiving lines in SCS (SNA [System Network Architecture] Character Stream), or IPDS (Intelligent Print Data Stream) format, and converting data fields in such lines into respective fields in AFPDS (Advanced Function Print Data Stream). Both SCS and IPDS have structured formats." **Dr. Birnbaum** discusses (¶¶15-23) the input data streams of the IBM prior art. **Dr. Birnbaum** states "The SCS data stream consists of a one-byte hexadecimal code followed by the data to be printed. SCS is used to control line printers and supports row and column functions. **Dr. Birnbaum** (¶17) asserts "...IPDS has no grammar and

no syntax across commands." At ¶18, **Dr. Birnbaum** opines, "It should be noted that page description languages (PDL) do not have "input print lines" and therefore, the statement in the IBM reference about "parsing" is not applicable to PDL input data streams, including Postscript. There is absolutely no disclosure in IBM on whether (and, if so, how) PDL inputs are "parsed" within any reasonable understanding of the term as used in the patent claims." **Dr. Birnbaum** opines (¶¶19-22) that the IBM conversion process of an input print line can be done by if-then statements, typical of a state machine and the '789 patent distinguishes its parser from a state machine. **Dr. Birnbaum** opines (¶22) on the '789 parser as "...complex... requires scanning the input data stream character by character, identifying tokens, e.g., collections of characters that correspond to commands or data and a syntax evaluation to determine the relationships between the identified tokens." "...IBM AS/400...cannot parse them into graphically representable objects. This is because the only methodology disclosed in IBM requires conversion into structured fields. (See e.g., IBM p. 194: input print lines are "parsed into individual fields")." **Dr. Birnbaum** opines (¶23), "IBM therefore does not disclose a parser as defined and claimed in the '789 patent. The cited portion of IBM only discloses conversion from structured SCS or IPDS format to AFPDS. That function is not parsing under any reasonable interpretation of the '789 patent claims, because it does not require syntax analysis."

Third Party Requester (Requester Comments 09/06/2011, pp. 3-5) citing to the language of claim 1 asserts the input print data stream is analyzed by means of a parser and IBM teaches (p. 194) that "[i]nput print lines are read in, optionally parsed into individual fields, and place'[d] on the page." **Requester** asserts (p. 3, and Jacobs Dec. ¶10) that Patent Owner arguments (p. 6-12

and Birnbaum ¶12) are not applying the broadest reasonable construction of the claims. "None of the claims recite a "syntax analyzer," a "formal grammar," or a "page description language." **Requester** asserts that such an approach improperly incorporates limitations from the specification to narrow this plain language. **Requester** asserts (p. 3) that the broadest reasonable interpretation of this language includes input data streams that are complex page description languages (and are parsed through syntax analysis) and input data streams that are structured fields (and can be parsed in other ways). **Requester** notes (footnote 1, p. 3) that the '789 patent provides little, if any, description about syntax analysis and Patent Owner to rely on declarant Birnbaum to describe and emphasize the alleged importance of this feature. **Requester** opines (p. 4) that IBM's SCS and IPDS data streams do have a syntax. **Requester** cites to Jacobs ¶10 and asserts "The existence of structured fields, and the ability to parse using a state machine, are not determinative of whether a data stream has syntax...the disclosure of structured fields in IBM's SCS and IPDS data streams is entirely consistent with its teaching of parsing." **Requester** notes (p. 4) that the alleged requirement of the parser of claim 1 to be a syntax analyzer is meaningless in view of claim 77. **Requester** makes note (p. 4) of an allegedly irrelevant comment (PO Remarks at 9, citing Birnbaum ¶21). Patent Owner is not arguing that SCS and IPDS print data streams cannot be parsed, only that there are ways to analyze the data streams in addition to parsing. **Requester** asserts that under the broadest reasonable interpretation that IBM clearly and explicitly teaches parsing an input print data stream. **Requester** asserts (pp. 4-5) that IBM (p. 14) discloses: "Printed output is the result of the interaction between the printer and the software that controls it. Different printed output requires <u>different types of printers and different software (data streams) to control them</u>." A print data stream is a set of commands that tell a

printer what to do. Some data streams, such as SNA Character String (SCS), include a limited number of commands. Others, such as Intelligent Printer Data Stream (IPDS), include a rich and complex set of commands."

Requester asserts (p. 5), "...IBM teaches receiving and analyzing PostScript print data streams. *(Id.* at 16.) PostScript is a page description language (PDL). *See* '789 Patent 2:52-53. IBM teaches receiving and analyzing PDLs, and the PO argues that analyzing a PDL requires a syntax analyzer parser. (Mr. Birnbaum, Patent Owner's declarant argues that due to their complexity, analyzing PDLs requires a syntax analyzer parser, Birnbaum ¶¶12-13.) Therefore, IBM teaches parsing even under the PO's narrow definition. To address Patent Owner's comment (PO Remarks at 9), that the "IBM AS/400 can handle Postscript documents as an input," but that "it cannot parse them into graphically representable objects," **Requester** opines that this is clearly not the case. "As stated in the OA (at 7-8) [Non Final Office Action 11/19/2010] and discussed in the next section, the IBM AFP architecture is an object-oriented architecture with graphic objects. IBM never teaches or suggests using a different (or non-object-oriented) architecture for use with PDL data streams."

The Jacobs Declaration (04/26/2011), in support of Third Party Requester, provides an opinion of one of ordinary skill in the art at the time period at and prior to May 11, 2001. **Dr. Jacobs** opines (¶9), "The Advanced Function Presentation/Printing (AFP) architecture described in the IBM reference describes receiving many different types of print data streams, including streams

referred to as "SCS" and "IPDS" as well as "PostScript." (IBM at 14-16.) IBM clearly teaches that one or more of these streams is "optionally parsed," as that term is commonly used in the art. (IBM at 194.) The ordinary meaning of "parsing" is not limited to syntactic parsing using a formal grammar or similar restriction. I understand that the PO argues that parsing in the '789 Patent claims means "a syntax analyzer: it is capable of analyzing the syntax of an input data stream based on a formal grammar," and that SCS and IPDS streams do not require that type of parsing. (PO Remarks at 6.) Even under the PO's construction, however, there is no reason why IPDS and SCS print data streams are not parsed, as recited in the claim.

Dr. Jacobs opines (¶10), "Birnbaum's analysis that IBM's SCS and IPDS data streams do not have a syntax is not correct. (Birnbaum Dec. ¶¶12-23.) The existence of structured fields and the ability to parse using a state machine are not determinative of whether a data stream has syntax. In fact,

parsers are frequently used to separate commands with structured fields and state machines have been used for parsing even complex languages such as English. Birnbaum also draws his conclusion because processing print lines "can be done by a series of if-then statements." (Birnbaum Dec. ¶20.) In fact, a series of if-then statements could be sufficient for even the most complex parsing tasks.

Examiner affirms that claims are given the broadest reasonable interpretation, consistent with the Specification (*Trans Tex. Holdings*). The best evidence for defining the scope of the

patented invention is the words of the claims themselves. These words should normally be given their ordinary and customary meaning unless the patentee has acted as its own lexicographer. Thus, the second best evidence, the specification, must always be consulted to at least determine if the terms were used in a manner inconsistent with their ordinary meaning. **Examiner** opines that the term "parser" was not used in a manner inconsistent with the ordinary meaning. **Examiner** agrees with the Jacobs declaration that the ability to parse using a state machine is not determinative of whether a data stream has syntax. Examiner maintains that the IBM reference fairly teaches "an input print data stream is read in...analyzed by means of a parser..."

Patent Owner asserts (Patent Owner Remarks, 03/30/2011, p. 9-12) that the IBM reference does not disclose storing graphically representable objects in an object-oriented format. Patent Owner cites to the Birnbaum declaration ¶¶24-26 and '789 at 3: 38-4: 8 to define the term "object-oriented format." **Patent Owner** notes the advantages of object class hierarchy at '789 7: 24-30 and 4: 40-46, with an example at 4: 50-61.

Patent Owner asserts (p. 11) that "Samsung purports to find "object-oriented architecture" at IBM Reference, pp. 21, 23, and Fig. 5. However, the IBM Reference uses the term "object" in a wide variety of ways, as exemplified by the thirteen definitions of the word in IBM's IBM Dictionary of Computing, 10th Ed., 1993; copy at Appendix C [received 03/30/2011]. The IBM dictionary separately defines an "object" in the context of object-oriented design as distinguished from its use in connection with the AS/400: object... (10) In object-oriented design or programming, an abstraction consisting of data and the operations associated with that data... (11) In the AS/400 system, a named storage space that consists of a set of characteristics that

define itself and, in some cases, data. An object is anything that exists in and occupies space in storage and on which operations can be performed; for example, programs, files, libraries, and folders."

"The IBM Reference uses the term "object" to refer to data structures in which various document elements (e.g., image, form, text, font, bar code, and graphic) are represented. For a number of reasons -- including lack of: class hierarchy, assignment of methods to objects and inheritance -- "object," as used in the IBM Reference, does not have the same meaning as "object" in object-oriented programming, and as that term is used in the '789 patent. (Birnbaum ¶27). Indeed, the structured field format of AFPDS may be suitable for storing such data elements, but it is not suitable for storing objects in object-oriented format. (Id.)"

"So, for example, page 23 of the IBM Reference, purports to describe as an "object" an image of an airplane. However, that so-called "object" is not handled as a member of a class to which dedicated methods are assigned - and therefore cannot be considered an object in the object-oriented programming sense. (Birnbaum ¶28) Moreover, the IBM Reference uses Mixed Object: Document Content Architecture-Presentation (MO:DCA-P) (IBM, p. 21)."

"IBM has defined a single object-oriented data stream--Mixed Object Document Content Architecture (MO:DCA). (An object is a collection of data that can be treated as a unit.) (Emphasis added). See, Birnbaum Dec. Exs. B 495 and C at 558." "The IBM Reference discloses simple data structures whose data are treated as a unit – and nothing more. (Birnbaum ¶29). By contrast, an object-oriented format, in the sense of the '789 patent and elsewhere, requires that objects be capable of being organized in hierarchical classes, where lower-class

objects inherit properties (including methods) from higher-class objects from which they descend. (Birnbaum ¶30)."

Patent Owner opines (p. 12), "When all the evidence is carefully considered, the IBM Reference does not disclose graphically represented objects stored in object-oriented format, a limitation in all claims of the '789 patent."

The **Birnbaum Declaration** (03/30/2011, ¶¶24-30) opines on the term "object-oriented programming": "object-oriented programming" uses objects - i.e., data, properties and methods, together with their interactions as defined by the methods...techniques typically include features such as class hierarchy, data abstraction, encapsulation, modularity, polymorphism, and inheritance." "An object-oriented format in the sense of the '789 patent, and as generally understood in computer programming, requires that objects have the ability to be organized, for example, in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend."

"IBM uses the term "object" to refer to data structures in which various document elements (e.g., image, form, text, font, bar code, and graphic) are represented." **Dr. Birnbaum** opines (¶27), "For a number of reasons -- including lack of: class hierarchy, assignment of methods to objects and inheritance -- "object," as used in IBM, does not have the same meaning as "object" in object-oriented programming, and as that term is used in the '789 patent." As support for that opinion, **Dr. Birnbaum** notes that IBM (p. 23) describes an "object" that is an image of an airplane. "However, that "object" is not handled as a member of a class to which dedicated

methods are assigned - and thus cannot be considered an object in the object-oriented programming sense." "IBM discloses simple data structures whose data are treated as a unit - and nothing more. (See, e.g., Ex. B at p. 495; Ex. C at 558; all received 03/30/2011). The structured field format of AFPDS is suitable for storing such data elements, but not objects in object-oriented format." **Dr. Birnbaum** opines (¶30), "By contrast, an object-oriented format, in the sense of the '789 patent and elsewhere, requires that objects be capable of being organized in hierarchical classes, where lower-class objects inherit properties (including methods) from higher-class objects from which they descend."

The **Birnbaum Declaration** cites to Exhibits B (IBM eServer iSeries Printer Device Programming v5, September 2002, p. 495) and C (AS/400e Printer Device Programming v4, May 1999, p. 558): "Mixed Object: Document Content Architecture (MO:DCA) The ability to print documents with consistent output, independent of either operating system or printer, is extremely important to the user of printed data. In order to help achieve this goal, IBM has defined a single object-oriented data stream--Mixed Object Document Content Architecture (MO:DCA). (An object is a collection of data that can be treated as a unit.) This architecture has been developed in order to meet several objectives: • The requirements relating to document and data sharing specified in IBM's Systems Application Architecture • Co-existence and migration of existing IBM document architecture and printer data streams • Device independence • Separation of functions to simplify transformation of objects into other data streams • National Language Support • Office Document Architecture (ODA) support..." From Exhibit B, p. 495, "The data stream for an MO:DCA document consists of various objects, such as text, images,

and graphics, as well as the logical and layout structure of the document. The logical structure defines the logical content of the document---chapters, figures, and lists. The layout structure defines the way the data should be presented."

Third Party Requester (Third Party Comments 09/06/2011, pp. 5-7) asserts that IBM's objects are hierarchical. See IBM at p. 218, Figure 129 and related text, "Applications that use the AFT Toolbox library functions need to follow the AFP hierarchy when building compound documents ...made up of objects..." **Requester** cites to the Jacobs Declaration at ¶11. **Requester** references the IBM dictionary definition (relied upon by Patent Owner) term "object" as used in "object-oriented" design or programming, is "an abstraction consisting of data and the operations associated with that data."

The **Jacobs Declaration** at ¶11 repeats the details as found in Requester paragraph above. Additionally, **Dr. Jacobs** evidences the inheritance features of the IBM objects (Ex. V, at 10; received 09/06/2011): "Mixed Object Document Content Architecture", which describes the data stream as used by AFP, as referenced in the IBM reference. Selected documents are included in Exhibit V, and show that the IBM objects support inheritance. "Hierarchical Defaults: Parameter values established by an environment group at a higher level in the document component hierarchy will be the default for a subordinate level unless a value is specified at the subordinate level .... (organized in hierarchical classes) The placement of parameter values at a higher level in the document hierarchy, for the purpose of enabling lower levels to <u>inherit</u> these values as

defaults, is known as factoring." (emphasis added)

Examiner notes that Birnbaum's Exhibit B, p. 496-497 further describes the objects of IBM: "...Specifies fonts, overlays, and segments so that these objects (graphically representable objects) can be transmitted as part of the data stream. They can be referenced by an MO:DCA include structured field." "Pages Contains objects that are part of the document. These objects could be text, graphics, and images." (graphically representable objects) "The following different types of objects make up MO:DCA. All of these objects are supported by IPDS: • Bar Code Object Content Architecture (BCOCA) • Image Object Content Architecture (IOCA) • Graphics Object Content Architecture (GOCA) • Presentation Text Object Content Architecture (PTOCA) • Font Object Content Architecture (FOCA)." (emphasis added)

"Image Object Content Architecture (IOCA) IOCA represents images in device-independent format. A standard set of constructs has been defined to describe the image data, the characteristics of that data, and manipulation functions that may be performed on the data. The image content is inserted in an image segment." "Graphics Object Content Architecture (GOCA) GOCA describes complex pictures. These pictures are formed from a collection of primitives, such as lines, arcs, characters, symbols, and shaded areas or point arrays. Each of these primitives has its own set of attributes, such as line width, orientation, and resolution. In addition to these attributes, there is a set of general drawing attributes like color, which apply to all primitives." (emphasis added)

"Presentation Text Object Content Architecture (PTOCA) PTOCA describes the text part of a document. The presentation text object, in common with the other objects, is designed not only to be carried by, but to be an integral part of, the data stream, providing the following: • Structured field introducer and syntax for the structured field • Begin/end object structure • Control of alternate action selection for error recovery • Passing of exception conditions back to the originating process • Initial state of the object • Relationship of presentation text objects to other objects contained in the data stream." (emphasis added)

"Two structured fields provide the necessary presentation information to the printer: **P T descriptor structured field** Defines several positional parameters for the object; **P T data structured field** Contains the presentation text and the control sequences for positioning graphic characters. These graphic characters are defined within the coded fonts."

Examiner asserts that IBM sufficiently discloses storing graphically representable objects in object-oriented format. Examiner strongly disagrees with Dr. Birnbuam's assertion (Birnbaum Declaration ¶27) referencing an airplane image (IBM at p. 23) as not handled as a member of a class to which dedicated methods are assigned. The cite evidences a page object that includes a form object, text object, font object, bar code object, image object, and graphic object. Notably IBM discloses manipulation functions (see paragraph above) associated with objects. The court standard is based on clear and convincing evidence standard to overcome presumption. The USPTO relies on a substantially lower standard, a preponderance of evidence.

Patent Owner asserts (pp. 12-13) that the IBM reference does not disclose use of a script. **Patent Owner** cites to the Birnbaum Declaration (¶¶31-38) for an interpretation of the term "script." **Patent Owner** asserts (p. 12) that the IBM Reference does not use the word "script", nor does it refer to DDS as "scripts"; "on the contrary, DDS must be compiled before being used..." "...the DDS described in the IBM Reference are not scripts as claimed in the '789 patent..."

The **Birnbaum Declaration** asserts (¶¶31-38): A script is a program or series of commands that is interpreted and runs in real time rather than compiled and then executed." "Scripts are defined as: "[a] high-level programming, or command, language that is interpreted (translated on the fly) rather than compiled ahead of time." And, "[s]cripts are interpreted as they are run." See Exhibit D." "In contrast to scripts, programs written in standard program languages, such as FORTRAN, C++, etc., must typically be compiled into object code before being executed." "Using scripts provides at least two advantages: flexibility and portability." "Scripts may be freely modified, including at run-time, which is not true for compiled code. For example, in one embodiment of the invention, the data processing unit permits stored objects, including particularly script objects, to be read out graphically, to be changed, to be deleted or to be appended. See, e.g., claims 18, 19. This cannot be done using compiled object code." "Object code is limited because it is platform-dependent. A program written in a compiled language must be compiled into object code for a particular processor architecture. In contrast, any processor capable of interpreting a script can run the script. This renders scripts portable, i.e., platform-independent." "This means the DDS in IBM is compiled before being used for priming, can only be executed as

object code after being compiled, and, unlike scripts, must be recompiled if modified." "This feature of DDS is a critical distinction from a script, in which interpretation and execution of the script are typically simultaneously triggered, in the case of the '789 patent, for example, by the incoming print data."

Dr. Birnbaum's Exhibit D (IBM Terminology-terms and definitions from many IBM software and hardware products, 09 December 2010; received 03/30/2011) evidences a definition of "script": (1) A series of commands, combined in a file, that carry out a particular function when the file is run. Scripts are interpreted as they are run. (2) The logical flow of actions for a 3270 server program. (3) An exact text for the telesales service representative to read to a customer regarding transactions. Scripts can be short-hand or prompts to remind a representative to say certain things to a customer at certain points during a call.

Third Party Requester (Third Party Comments 09/06/2012, pp. 7-8) asserts that IBM (pp. 127-128) teaches scripts referred to as "Data Description Specification" or "DDS" commands. **Requester** opines that the broadest reasonable interpretation of scripts, in light of the '789 patent specification, does not preclude compilation. "...the '789 patent provides an example script called "JavaScript." (See e.g., 6:52-53; claim 19.) **Requester** evidences scripts as compilable. (Ex. R, Server-Side JavaScript Guide v4.0 Sun Microsystems, 1999 at p. 39; received 09/06/2011; Jacobs Dec. at ¶¶12-13): "If the HTML and JavaScript files contain server-side JavaScript, you then compile them into a single JavaScript application executable file. The

executable is called a web file and has the extension .web. The JavaScript application compiler turns the source code HTML into platform-independent bytecodes, parsing and compiling server-side JavaScript statements." **Requester** opines (p. 7), "If Javascript - the scripting language recited in the '789 patent - can be compiled, then the compilation or non-compilation of the script is clearly not a requirement of scripts."

Requester additionally notes (p. 8), "The IBM reference itself does not describe whether DDS commands are compiled. Instead, the PO refers to a later document called the "IBM iSeries Printer Device Programming, Version 5" (hereinafter, the "iSeries Version 5" document). "This argument fails for several reasons. For one, reexamination is limited to published prior art references, not prior use. The question before the Examiner is whether the claims are patentable over the IBM reference used in the reexamination, not over the IBM iSeries Printer. Second, this second IBM document is dated 2002 and is not prior art. Third, the second document is not directed to the same version of the IBM iSeries Printer. The IBM reference used in the reexam refers to Version 4; the second document cited by the PO refers to version 5. All this reference teaches is that in a later version of the printer, DDS was compiled."

The **Jacobs Declaration** at ¶¶12-13 further asserts that script compilers were commonly used at or before May 11, 2001. **Dr. Jacobs** asserts (¶13), "It is a well-known principle that compiled scripts often run faster than non-compiled scripts, but non-compiled scripts can run on more systems. People of ordinary skill in the art routinely choose whether to use compiled or non-compiled scripts, dependent upon their specific situation. I see no reason why any aspect of a

script described in the '789 patent claims requires non-compiled scripts, instead of compiled scripts."

Examiner asserts that the term "scripts" is not narrowly construed so as to preclude compiling. **Examiner** takes exception to Dr. Birnbaum's implication that scripts are only interpreted as they run. Compiled DDS commands fairly read on the term "script." The '789 specification teaches (6: 10-11) executing a Java Script at 6: 52-53 expressly states that the preferable language used for the scripts is Java Script. **Examiner** agrees with Requester that Java Scripts may be compiled. The '789 specification describes JScribe sequences as scripts with appropriately associated objects and otherwise is silent as to whether they are interpreted only, or compiled. Patent Owner does not cite to a narrow definition of the term "script." Claims 17 and 18 do not preclude compiled scripts. As broad support for compiled scripts, see '789 7: 47-52: "The application interface also preferably permits script objects, preferably Java Script objects themselves, to be read out graphically, to be changed...these graphically performed manipulations being automatically transformed, if required, into script objects, preferably Java Script objects." (emphasis added)

See IBM, p. 127: "DDS printer file support provides full control of output, and supports the advanced electronic printing capabilities of today's laser printers. DDS gives you complete control over each page and over all the elements that come together on a page." "The printer file, through the DDS keywords, controls the position, orientation, font, and other characteristics for those fields. In addition, DDS provides access to all the elements--text, overlays, images,

graphics, bar coding, lines, and boxes--that comprise AFP documents." Dr. Birnbaum's definition, "A series of commands, combined in a file, that carry out a particular function when the file is run" is consistent with IBM's teachings.

Examiner asserts that while the iSeries Version 5 document does qualify as a published document, because it is dated 2002, Examiner agrees with Requester that it does not qualify as prior art and is not directed to the same version of the IBM iSeries Printer.

Patent Owner asserts (p. 13) that the IBM reference does not disclose a script assigned to an object. Patent Owner repeats that the images, text, etc. elements in the IBM reference are not objects stored in object-oriented format, DDS is not a script, and DDS is not assigned to an element, but rather applied to the document as a whole. **Patent Owner** cites to the Birnbaum Declaration ¶¶39-41. **Patent Owner** argues (p. 13) that the benefits of a script assigned to an object include automatically sending / requesting/ receiving data, reassigning data received to a graphic object, forwarding the graphic associated and asserts that the AS/400 machine cannot perform these functions.

The **Birnbaum Declaration** (¶¶39-41) opinion align with Patent Owner's remarks: "Even assuming that the "elements" in IBM (images, text, etc.) were objects stored in object-oriented format, and that the DDS used in IBM is a script, the DDS is not assigned to an element, but rather is applied to the document as a whole." "As described in the '789 patent, by assigning a

script to a particular object, various benefits and advantages may be obtained. (See e.g., col. 5, lines 27-31 and col. 6, lines 38-51)." "The architecture of the AS/400 described in IBM would have to be entirely reconfigured and the code totally rewritten to perform these operations."

Third Party Requester (pp. 8-10) disagrees with Patent Owner and Dr. Birnbaum. "IBM (p. 128) teaches that the assignment of DDS keywords (scripts) can be at the record level (document) or at field level (objects)." (IBM at 128.) "The "record-level" DDS commands are assigned to an entire print stream, whereas the "field-level" DDS commands are assigned to individual objects. IBM further explains: The printer file, through the DDS keywords, controls the position, orientation, font, and other characteristics for those fields. In addition, DDS provides access to all the elements (objects)--text, overlays, images, graphics, bar coding, lines, and boxes--that comprise AFP documents." (IBM at 127.) **Requester** cites to the Jacobs Declaration (¶14) as an example of a script assigned to a text object. **Requester** opines that Patent Owner reliance on a number of functions as necessarily embodied in a script assigned to an object is improperly arguing limitations (of dependent claims 5-11) that were not claimed (in claim 1).

The **Jacobs Declaration** (¶14) evidences DDS commands assigned to specific objects. See IBM at 127-127; Fig. 63, The TXTRTT script, text rotate (script) command applied to text objects.

Examiner asserts, as noted above, that a DDS command fairly reads on a "script." **Examiner** disagrees with the Birnbaum declaration that scripts are only applied to a document as a whole, and additionally points out that a document also is an object. **Examiner** agrees that a TXTRTT command fairly reads on a script assigned to a graphically representable object, i.e., a text object. The various functions, as asserted by Patent Owner, that **may** be assigned by a script to an object are not recited in the claim language. The argument is not on point with claim language.

Regarding **Ground of Rejection #2,** claims rejected as obvious over the IBM reference in view of Interleaf. See Request pp. 5, 7-8, and Exhibit H / IBM-Interleaf Claim Chart; all received 07/16/2010. **Patent Owner** asserts (pp. 14-20) that there is no prima facie case of obviousness. **Patent Owner** assesses one of ordinary skill in the art (presumed to be one who thinks along the line of conventional wisdom in the art and not one who undertakes to innovate), noting that this person does not look for exceptionally creative solutions or to completely redesign existing software. **Patent Owner** asserts (p. 15) that the field of the invention is non-analogous to document creation or user interfaces, as disclosed by Lieberman. **Patent Owner** asserts that the field of the invention of the '789 patent is printing software generally and in particular interpretation or processing of print data streams at a printer or print server. **Patent Owner** cites to the Birnbaum Declaration ¶¶53-56.

Patent Owner opines (pp. 15-16) that IBM and Interleaf references are in entirely different fields and a person of ordinary skill in the art would not have combined them. **Patent Owner**

summarizes the prior art and cites to the Birnbaum Declaration ¶¶58-61. **Patent Owner** asserts (p. 16-18) that hindsight rephrasing of keywords from the '789 patent are relied upon by Requester (in claim charts Exhibit H -claim chart for IBM in view of Interleaf, p. 11 and Exhibit J (claim chart for Interleaf alone and for the combination of Interleaf and Lieberman), p. 21; received with Request 07/16/2010) to identify two fields of the invention: object oriented document publishing systems with scripting functionality and scripting interfaces for object systems.

Exhibit H, p. 11: A person of ordinary skill in the art would have reason to combine IBM and Interleaf because they both address the same field, namely object-oriented document publishing systems with scripting functionality. The above-described combination of prior art elements is done according to known methods to yield predictable results. *See* MPEP 2141 .III.(A).

Exhibit J, p. 21: A person of ordinary skill in the art would be motivated to combine Interleaf with the Interleaf Patent and Lieberman because they both address the same field, namely scripting interfaces for object systems. This overlap in the field of use would motivate a person of ordinary skill in the art to combine Interleaf, the Interleaf Patent, and Lieberman.

Patent Owner cites (p. 16) to Appendix D (03/30/2012, Patent Owner generated comparison chart of patent classifications) evidencing the USPTO patent classifications. **Patent Owner** asserts (p. 18; and cites to the Birnbaum Declaration ¶64), in reference to an invention combining IBM and Interleaf teachings, that "any object oriented formatting, including any scripts, in the Interleaf 'active document' would be lost prior to generating the print data stream

that would be transmitted to the AS/400.". **Patent Owner** asserts (p. 18) that there is no suggestion to redesign the system, arguing that such modification would change the principle of operation of the prior art being modified. **Patent Owner** opines (p. 18) that IBM teaches the desire to control outside printing and publishing functionality. The external functionality of Interleaf refers to printing, whereas the external functionality of IBM refers to staplers and binders, neither of which requires scripting functionality or object oriented code. **Patent Owner** asserts that "common words (without a common meaning) cannot...be the basis for a motivation to combine the references. **Patent Owner** asserts (pp. 19-20 and cites to Birnbaum Declaration ¶¶67-72) that the combination of IBM and Interleaf References would not have resulted in the inventions of the '789 patent: "[n]othing in the print stream is changed: there still are no objects, no scripts assigned to objects and no parser."

Patent Owner asserts (pp. 20-24) secondary considerations and commercial success, which are addressed at the beginning of the arguments.

Patent Owner does not cite to the **Birnbaum Declaration** ¶¶42-44, but the Birnbaum opinions presented reference the printer of claim 20 (of the system of claim 17), noting that IBM's AS/400 is not a printer, nor would a printer connected to the AS/400 have a system for the transformation of digital print data streams, nor would such a printer have the capability to compile or execute DDS. Dr. Birnbaum asserts that it would not have been obvious to take the functionality of the AS/400 and import it into a printer. Patent Owner does not cite to the

Birnbaum Declaration ¶¶45-52, but it is noted that Dr. Birnbaum asserts that limitations of dependent claims are not suggested by IBM.

The **Birnbaum Declaration** (¶¶53-61) opines on "one of ordinary skill." **Dr. Birnbaum** asserts (¶58) that IBM describes how data may be formatted and sent to a printer and print capabilities of networks controlled by a central server, like the AS/400. **Dr. Birnbaum** asserts (¶59-61) that Interleaf (p. 75) deals with 'document creation' and manipulation software. "...the Interleaf documents do not involve printer software or interpretation of the print data streams at a printer of print server." Opining that the Interleaf documents are in an entirely different field of technology than the '789 patent, "One of ordinary skill in the art would not have referred to the Interleaf documents or combined them with IBM to arrive at the inventions of the '789 patent." The **Birnbaum Declaration** (¶¶62-72) argues the motivation to combine repeating the same comments as Patent Owner.

Third Party Requester asserts (pp. 10-15) that "IBM, Interleaf, and the Interleaf Patent are in the same field of invention or endeavor, regardless of any different classifications for the '789 patent and the Interleaf Patent. The claims of the '789 patent recite processing steps that are performed in a computer, with the last step being outputting objects to "an output device, preferably a printer." IBM, Interleaf, and the Interleaf Patent are also directed to processing data in a computer, with the last step being providing the processed data to a printer. The PO does not contest that IBM is in the same field of endeavor as the '789 patent, but rather contests

whether Interleaf and the Interleaf patent are directed to this field." "...Interleaf describes a system that receives a "document file stream", transforms the stream into "active" documents, and results in the "delivery of active documents." *(See, e.g.,* OA at 27-29; Interleaf at 75.) Interleaf further teaches (p. 80) that the "active objects within the document file stream ... become activated when the document is opened programmatically or by the user for viewing, editing, or printing." "The PO focuses on the fact that Interleaf allows the user to view or edit the file stream, but ignores the fact that the file stream can also be opened programmatically (like the '789 patent) and/or be used for printing (also like the '789 patent). Also, and in further contradiction to the PO's argument, claim 18 of the '789 patent recites "viewing" and "editing." " "Furthermore, the Interleaf Patent states: "This invention relates to a Document Processing System for creating, editing, printing and presenting active electronic documents which are associated with computer programs." (1:8-11) Even if the Interleaf Patent was classified according to the "creating" and "editing" aspects of the disclosure, this does not negate the fact that the Interleaf Patent is also directed to "printing" active documents." "Further still, even if the references were in different fields, that does not prohibit an obviousness combination from being found. The key to supporting any rejection under 35 U.S.C. 103 is the clear articulation of the reason(s) why the claimed invention would have been obvious."

Requester makes note of the '789 patent, 8:40-58; 9:39-39 (claim 1); 11:30-34 (claim 22); 13:9-13 (claim 38) and states, "...claims do not require the output be provided to a printer, only preferably." Requester asserts (p. 10 footnote) IBM (p. 1) describes a printing system: Advanced Function Presentation (AFP) is a printing and presentation system that enables you to take full advantage of printing technology on your AS/400 system. AFP helps you combine the

capabilities of high-function Intelligent Print Data Stream (IPDS) printers and print software to: • Create state-of-the-art documents • Exploit print formatting capabilities without changing application programs • Replace traditional, labor-intensive print operations with a system-managed process • Print with complete system management and full error recovery, whether the printer is system-attached via twinax or network-attached via TCP/IP.

Requester notes that Interleaf describes (Interleaf at 75, 84, emphasis added) delivering active documents for printing: A commercial structured document processing system [that] has been built with an extensible object system. This system is an excellent platform for the design, implementation, and delivery of active documents ...active behavior occurs in these ways: "... If a document contains active objects within the document file stream (as opposed to the methods file), these become activated when the document is opened programmatically or by the user for viewing, editing, or printing." (emphasis added)

Requester cites (p. 12) to the MPEP and KSR for justification to combine: MPEP 2141.01(a) states: a reference in a field different from that of applicant's endeavor may be reasonably pertinent if it is one, which, because of the matter with which it deals, logically would have commended itself to an inventor's attention in considering his or her invention as a whole. *KSR* states: When a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a person of ordinary skill can implement a predictable variation, § 103 likely bars its patentability. For the same reason, if a technique has been used to improve one device, and a person of ordinary skill

in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond his or her skill. *KSR Int'l Co. v. Teleflex Inc.,* 127 S. Ct. 1727, 1740, 82 USPQ2d 1385, 1396 (2007), emphasis added.) In the present situation, even if the Interleaf patent was in a different field of endeavor, it teaches a technique that has been used to improve an active document system, and would similarly approve printing systems in the same way, with similar, predictable results.

In response to Patent Owner's argument that the prior art devices are not physically combinable, **Requester** cites to *KSR Int'l Co. v. Teleflex Inc.,* 550 U.S. 398, 420-421 (2007), noting that such an argument is not sufficient to overcome an obviousness rejection: "Common sense teaches, however, that familiar items may have obvious uses beyond their primary purposes, and in many cases a person of ordinary skill will be able to fit the teachings of multiple patents together like pieces of a puzzle .... A person of ordinary skill is also a person of ordinary creativity, not an automaton.") and *In re Icon Health & Fitness, Inc.,* 496 F.3d 1374, 1382 (Fed. Cir. 2007)("we do not ignore the modifications that one skilled in the art would make to a device borrowed from the prior art.") **Requester** asserts (p. 13) that it would be obvious for one of ordinary skill in the art to combine IBM with the scripts of Interleaf because scripts were known tools for printing, as evidenced by both IBM and Interleaf. IBM clearly uses objects, and it was well known to a person of ordinary skill in the art that scripts can enhance the functionality of objects. Running the scripts of Interleaf on the IBM system would perform the same, predictable function that they run on the Interleaf system. **Requester** cites to the Jacobs Declaration at ¶¶14-15 for support. **Requester** agrees with Patent Owner concerning advantages and disadvantages of using

compiled code versus non-compiled code, but further asserts that both are well known in the art, and choosing between the two would be a mere choice of design for someone of ordinary skill in the art. (Jacobs ¶¶12-13.) For support **Requester** cites (p. 13) to The Federal Circuit: [A] given course of action often has simultaneous advantages and disadvantages, and this does not necessarily obviate motivation to combine. The fact that the motivating benefit comes at the expense of another benefit, however, should not nullify its use as a basis to modify the disclosure of one reference with the teachings of another. Instead, the benefits, both lost and gained, should be weighed against one another. *(Medichem, S.A., v. Rolabo, S.L.,* 437 F.3d 1157, 1165 (Fed Cir, 2006), citations omitted.)

Additionally **Requester** finds (p. 13) that IBM broadly states that: "AFP Data Stream is independent of operating systems and page printers, [and] is portable across environments[.]" (IBM at 15; *see also,* 265-69, which states that the IBM system works with many different printers, including HP and Lexmark printers.) This shows that even if the DDS scripts are compiled, IBM still has achieved the benefit of platform independence. **Requester** opines (p. 13) that Patent Owner's argument wrongly assumes - contrary to the requirement that claim terms be given their broadest reasonable construction - that a script must be non-compiled. (See Ex. R, Server Side Java Script Guide v4.0, 1999; received 09/06/2011, pp. 7-8, regarding compiling JavaScript.)

Requester responds (pp. 13-14) to Patent Owner argument (Patent Owner Remarks, p. 18) regarding "the 'external' functionality of Interleaf refers to printing, whereas the 'external'

functionality of IBM refers to staplers and binders, neither of which requires scripting functionality or object oriented code" by citing the teachings of '789 patent at 5: 3-10 and 6: 47-51 (at least one graphically representable object stored in the memory in the object-oriented format is assigned at least one script which controls external devices, preferably archiving devices, folding systems, enveloping systems, or security equipment, which permits the incorporation of all the devices needed in the widest sense of document processing). **Requester** notes (p. 14) that IBM teaches a folding script (ZFOLD, IBM at 134-135), which is just like the "folding systems" described in the '789 patent.

In response to Patent Owner's argument that "[n]othing in the print stream is changed...", **Requester** asserts (p. 14) that these elements are taught by Interleaf and IBM, as shown in the original Request, and that Patent Owner has not addressed the rejections.

Requester addresses (pp. 14-15) secondary considerations and commercial success, which are discussed at the beginning of Response to Arguments above.

The Jacobs Declaration (¶¶ 12-13) disagrees with the (Patent Owner & Dr. Birnbaum) assertion that scripts must be non-compiled structures, noting that the '789 patent refers to different kinds of scripts, including scripts in the programming language "Javascript." Scripts written in many languages, including Lisp, Javascript, Perl, and Python are regularly compiled using either

ahead-of-time (AOT) or just-in-time (JIT) compilation. These script compilers were all in common usage at or before May 11, 2001. For example, see Exhibit R, "Server-Side JavaScript Guide" (p. 39) for a teaching of compiling scripts. **Dr. Jacobs** opines (¶13), “It is a well-known principle that compiled scripts often run faster than non-compiled scripts, but non-compiled scripts can run on more systems. People of ordinary skill in the art routinely choose whether to use compiled or non-compiled scripts, dependent upon their specific situation. I see no reason why any aspect of a script described in the '789 patent claims requires non-compiled scripts, instead of compiled scripts.”

Dr. Jacobs asserts (¶14) that the IBM reference also teaches that DDS commands are assigned to specific (noting IBM’s "record level" and "field level") DDS commands. Record level DDS commands are assigned to the entire document or print stream. Field level DDS commands are assigned to individual objects. (See IBM at 127-128; Fig. 63.) One example of a DDS script taught by IBM is the text-rotate script: "TXTRTT." (IBM at 134.) This script is assigned to one of the text objects *(see* text objects shown in Fig. 5 of IBM, reproduced above) so that the text prints at a different orientation. The result of assigning this DDS command to the text object "ROTATE" is shown in IBM Fig. 63...This command is clearly assigned to a particular object, because not all of the text or other objects in Fig. 63 are rotated.

Dr. Jacobs asserts (¶15) that the Interleaf reference, the Interleaf patent, and the Lieberman patent all refer to scripts, and the common usage of scripts in various software scenarios.

Scripting is a general-purpose, or multi-purpose software technique, and is not limited or restricted to one area of software technology. A person of ordinary skill in the art would be able to know and use scripts in many different situations. Consistent with this fact, the scripts described in the '789 patent, IBM, Interleaf, the Interleaf patent, and Lieberman all operate in a predictable manner. Using a type of script in one of these references in any other reference would function in its known, predictable manner. For example, the "Server-Side JavaScript Guide" shows examples of scripts also being used to manipulate files (Ex. R at pp. 170-177), send email messages (Ex. R at pp. 167-170), access databases (Ex. R at pp. 189-262), dynamically load functions and libraries (Ex. R at pp. 178-181), and other various functions. Exhibit S, U.S. Pat. No. 6,678,705; received 09/06/2011, has examples of scripts (in various languages, including Javascript) sending and receiving email (Ex. S at 3:8-4:30), retrieving information from Internet servers (Ex. S at 6:49-8:41), as well as interacting with various databases (Ex. S at 8:50-11:23).

Dr. Jacobs further provides opinions (¶¶16-18) regarding the obviousness of the combination of the cited prior art: Interleaf describes (p. 84) transforming a "document file stream" into active documents. The PO has made several arguments that a document file stream is not the same as a "print data stream," as that term is used in the '789 patent. I have been asked to consider this scenario, and whether it would be obvious for a person of ordinary skill in the art at the time of the filing of the '789 patent to use Interleaf technology with a print data stream (as shown in IBM) instead of a document file stream. The answer is yes, for several reasons. First, Interleaf and IBM both describe a computer system receiving a data stream, and performing the exact

same operations as done in the '789 patent. ('789 patent at 8:40-48.) The received data stream performs the same function in each reference. Both references also refer to performing various operations on the transformed data stream, as discussed in the OA. [Non Final Office Action 11/19/2010] Second, IBM lists "PostScript" as an example input print data stream. (IBM at 16.) It would have been obvious to use a PostScript data stream instead of a Lisp-oriented document stream--as Interleaf teaches--because the syntax of PostScript was based on and inspired by Lisp. *(See e.g.,* Ex. W at 1, "The first few chapters of this book help you put PostScript into perspective by comparing it to languages you probably already know, such as C. There are stronger similarities between PostScript and, say, Forth or Lisp, but if you are expert in either of those languages you probably won't have much trouble mastering PostScript.") [Exhibit W, received 09/06/2011, Reid, "Thinking in PostScript" 1990, p. 1]

Examiner agrees with Third Party Requester and the assertions and opinions of Dr. Jacobs. The example portions of motivation statements cited by Patent Owner (Exhibits H & J claim charts received 07/16/2010) are just a portion of entire motivation provided in the Office Action. **Examiner** does not see a contradiction in the commonality between the references and the patent under reexamination. Dr. Jacobs has evidenced the well-known usage of scripts in Exhibits R & S (received 09/06/2011).

Patent Owner arguments addressed to "external" functionality are not on point. Claim language "an output device, preferably a printer" does not require a printer be shown in the prior art.

Patent Owner clearly intended for the claim to be broad ('789, 5: 8-9, "...all the devices needed in the widest sense for document processing."). Running the scripts of Interleaf on the IBM system would perform the same, predictable function that they run on the Interleaf system. The '789 specification teaches a document data stream, which is identified as a print data stream. However the data stream may or may not actually print. The '789 specification teaches that the data stream may be edited to add script commands and saved to memory. Examiner addresses "hindsight" arguments at p. 140.

Regarding **Ground of Rejection #3**, obvious over the IBM reference in view of Interleaf and the Interleaf Patent. See Request pp. 10-12, and Exhibit I. **Patent Owner** asserts (p. 24) that the Interleaf Patent does not add relevant subject matter to the Interleaf reference. The Interleaf Patent, too, deals with document creation and editing. (Birnbaum Dec. ¶80). At most, the software described in the Interleaf Patent may generate a print stream, but it does not interpret or transform a print stream, as claimed in the '789 patent. Therefore, the same reasons, presented above, why the references would not have been combined, and in any event, would not have resulted in the inventions of the '789 patent if combined, are applicable here. The objective evidence of non-obviousness is likewise applicable.

The Birnbaum Declaration (¶80) opines that The Interleaf Patent does not add relevant subject matter to Interleaf. Both deal with document creation and editing. At most, the software described in the Interleaf Patent may generate a print stream, but it does not interpret or

transform a print stream, as claimed in the '789 patent. **Dr. Birnbaum** opines (¶¶81-93) the claims rejected under the combination of IBM, Interleaf and the Interleaf Patent:

Regarding claims 2, 23, and 39, the Interleaf Patent, at col. 2, line 61 to col. 3, line 13, bears no relation to transforming a print data stream...

Regarding claims 4, 25, and 41, the cited portion of the Interleaf Patent relates to a user-document interface, not a printing function. It is not clear at all how this relates to printing, or how the AS/400's DDS would have been capable of providing this functionality.

Regarding claims 5, 26, and 42, the cited portions of the Interleaf patent do not relate to use in connection with an input print data stream. For example, one application involves retrieving stock information and displaying it in text and charts; however, nothing suggests extracting the data at print time, that is, based on an input print data stream, much less doing so using the AS/400's DDS.

Regarding claims 6, 27, and 43, Interleaf and the Interleaf Patent relate to user editing and handling of documents, not their printing. Moreover, nothing suggests using the AS/400's DDS, to provide the functionality described in the Interleaf Patent, or that it is even possible to do so.

Regarding claims 7, 28, and 44, IBM does not disclose scripts, but merely the use of a compiled DDS on data. The portion of the Interleaf Patent that discloses a "customer receipt that automatically runs a 'frame grabber' and incorporates a photographic image of the purchaser" does

not transpire based on an input print data stream, but rather during creation of a document. Nor is there any indication that the DDS used in the AS/400 would have been capable of performing

such

functionality.

Regarding claims 8, 29, and 45, the cited portions of the Interleaf Patent have no relevance to a print application. Thus, for example, there is no reason to think that one of ordinary skill would have combined the references to produce an urgent memo that when printed integrates with voice annotation software so when it prints itself, it announces its presence audibly, or that this would have been possible using the tools available from IBM.

Regarding claims 9, 30, and 46, the disclosure of the Interleaf Patent for a "WYSIWYG document that can send itself over an electronic mail system" has no application to a transformation of a print data stream, as recited in the claims.

Regarding claims 10, 31, and 47, the Interleaf Patent's example of a "stock-market information center that retrieves stock information and displays it in text and charts that can be automatically distributed or incorporated in other documents" is in the limited context of document creation and editing. Once the document is sent to the printer, that functionality is lost. Nothing in the references suggests obtaining such information after receiving an input print data stream.

Regarding claims 11, 32, and 48, nothing suggests combining the features of the Interleaf Patent at col. 6, lines 6-16, and col. 3, lines 37-39 with IBM, which does not disclose scripts, but merely the use of a compiled DDS on data. Moreover, even if the Interleaf Patent discloses the functionality of these claims, it would not have been possible to incorporate this into the AS/400.

Regarding claims 12, 33, and 49, the portion of the Interleaf Patent that discloses a "document created from a database which updates the database if you make any changes in the imported

document" has no application to a transformation of a print data stream, as recited in the claims, nor does it seem possible to perform this functionality in the AS/400.

Regarding claims 13, 14, 34, 35, 50, and 51 the cited examples are not relevant to the inventions of the '789 patent. First, the examples relate to the generation of a document, not an operation performed on a print stream based on a document. Second, simply comparing an announcement date to a current system date is not a timer function. It is not at all clear it would have been possible to implement either the date comparison or the timer functions on the AS/400.

Regarding claims 16, 37, and 53, the various editing functions disclosed in the Interleaf Patent are not relevant to the '789 patent, because they are performed before generating an input print data stream. Nowhere does the Interleaf Patent disclose or render obvious that these functions are performed after receiving an input print data stream, as required by the claims.

Regarding claim 18, the cited functions of the Interleaf Patent occur before the objects are output in the output print data stream. The Interleaf Patent does not disclose performing them after receiving a print data stream.

Third Party Requester rebuts (p. 15), "Claims 1-14, 16-18, 20-35, 37-51, and 53 stand rejected as being obvious over IBM in view of Interleaf and the Interleaf Patent. The PO's remarks in this section merely reference their previous remarks, addressed above. This is insufficient to rebut the further disclosures included within Interleaf and the Interleaf Patent, and the manner in which the

combination of the IBM, Interleaf, and Interleaf Patent disclosures combine to show the various elements of the claims."

Examiner notes that the Interleaf Patent was added to the combination pointing out explicit teachings of (1:30-41) "at least one stored script is assigned" to the graphically representable objects, "... an improved document processing system for creating computer procedures (i.e., methods) and associating them with an electronic document structured as a nested hierarchy of constituent objects... means for defining the method and associating it with the electronic document or its lower level constituent objects....further includes <u>an interpreter for interpreting</u> the computer procedure in response to a specified event..." (3: 37-39), "As explained above, any data object within the hierarchy may be associated with a method which, upon the occurrence of a certain event, will be implemented." Interleaf Patent discloses specific examples that read on claim language. Interleaf and the Interleaf Patent are both directed to the same physical system. The existence of this system serves as an explicit suggestion to combine the teachings of Interleaf and the Interleaf Patent, which both describe the system. IBM (p. 404) is relied upon for teachings of external printing and publishing functionality. "Transformation of digital print data streams" is taught by IBM, as noted in the rejection and claim chart. See Non Final Office Action, p. 18 for entirety of motivation to combine. Regarding the argument that the Interleaf Patent deals with document creation and editing (Birnbaum Dec. ¶ 80) and may generate a print stream, but it does not interpret or transform a print stream, as claimed in the '789 patent, **Examiner** points out that the Interleaf document input data stream, document read in for processing / transformed is fairly analogous to a "digital print data stream" that is input,

transformed, and may or may not actually be sent to a printer output device. Both are digital. Both are data streams. Both may or may not actually be outputted to a print device.

See Non Final Office Action 11/19/2010, pp. 16-19 and Exhibit I; 07/16/2010, pp. 1-13 and in general the entire claim chart. Patent Owner does not directly argue the citations provided in the rejections. As rebutted above, interpreting script is not required by claim language.

In rebuttal to Dr. Birnbaum's statement (¶81) regarding claims 2, 23, and 39, **Examiner** notes that claim language does not recite "to transforming a print data stream", but rather recites "graphically representable objects are combined into super-objects of higher complexity before being stored in the memory." See Non Final Office action 11/19/2010, p. 19 for an explicit teaching by Interleaf Patent at 2: 61-3: 13.

In rebuttal to Dr. Birnbaum's statement (¶82) regarding claims 4, 25, and 41, **Examiner** cites to the Non Final Office action 11/19/2010: Interleaf Patent (10: 16-20) teaches a script controlling external devices. Controlling a printer, or a printing function is not required by claim language (claim 1 recites "preferably a printer)."

In rebuttal to Dr. Birnbaum's statement (¶83) regarding claims 5, 26, and 42, **Examiner** asserts that claim language does not recite "extracting the data at print time that is based on an input print data stream." The term "print time" is not disclosed and not germane to any argument.

Clearly the '789 method steps may never progress to "print time," as the modified data stream may be stored, or output to some other output device.

In rebuttal to Dr. Birnbaum's statement (¶84) regarding claims 6, 27, and 43, (Interleaf and the Interleaf Patent relate to user editing and handling of documents, not their printing...nothing suggests using the AS/400's DDS, to provide the functionality described in the Interleaf Patent, or that it is even possible to do so), **Examiner** notes that arguments are not directed to claim language (...the script automatically receiving data also requests this data automatically.) See motivation to combine prior art addressed in the rejection of claim 1, Non Final Office Action 11/19/2010, pp. 15-19.

In rebuttal to Dr. Birnbaum's statement (¶85) regarding claims 7, 28, and 44, **Examiner** notes that arguments are not on point with claim language. The argument is not clear.

In rebuttal to Dr. Birnbaum's statement (¶86) regarding claims 8, 29, and 45, **Examiner** asserts that claim language does not recite "a print application." Obvious reasons to combine references are rebutted above. See motivation to combine prior art addressed in the rejection of claim 1, Non Final Office Action 11/19/2010, pp. 15-19.

In rebuttal to Dr. Birnbaum's statement (¶87) regarding claims 9, 30, and 46, **Examiner** notes that arguments are not directed to claim language "...script sends the graphic object associated with itself to receiver." Regarding the argument related to "a transformation of a print data stream" a rebuttal above addresses the limitations as noted in claim 1.

In rebuttal to Dr. Birnbaum's statement(¶88) regarding claims 10, 31, and 47, **Examiner** asserts that claim limitations are addressed in the Non Final Office action 11/19/2010, p. 22. The

citation teaches stock market information retrieved and incorporated into a document. See claim chart, Exhibit L, Claim chart, 07/16/2010, pp. 26-27 for additional examples. The Interleaf Patent is directed to "printing." Claim language does not recite "obtaining such information after receiving an input print data stream."

In rebuttal to Dr. Birnbaum's statement (¶89) regarding claims 11, 32, and 48, **Examiner** notes that arguments do not address claim language. Arguments are directed to the motivation to combine Interleaf Patent and IBM which is discussed above.

In rebuttal to Dr. Birnbaum's statement (¶90) regarding claims 12, 33, and 49, **Examiner** notes claim language does not recite "transformation of a print data stream." See claim language fairly rejected in Non Final Office action 11/19/2010, and Exhibit I Claim chart, 07/16/2010, pp. 47-44. The limitation "transformation of digital print data streams" is found in the preamble of claim 1.

In rebuttal to Dr. Birnbaum's statement (¶91) regarding claims 13, 14, 34, 35, 50, and 51, **Examiner** notes claim language recites "...a timer...automatically as a result of expiry of time" & "...timer...starts itself again upon expiry" and such limitations are fairly disclosed by the Interleaf Patent (6: 32-53, "system clock reaching a certain time or date"). The Interleaf Patent teaches multiple examples of a clock with date or time awareness that triggers an action. See Exhibit I Claim chart, 07/16/2010, pp. 44-46. It would be obvious to implement such functionality into a script.

In rebuttal to Dr. Birnbaum's statement (¶92) regarding claims 16, 37, and 53, **Examiner** notes that rejection cites to Interleaf Patent, 1: 15-20, a document (read into a computing machine) is

edited on a workstation. There is no claim language reciting "generating an input print data stream." Claim 1 recites "an input print data stream is read in..." and IBM is relied upon for teaching the limitation. See Exhibit I Claim chart, 07/16/2010, pp.1-2.

In rebuttal to Dr. Birnbaum's statement (¶93) regarding claim 18, **Examiner** notes the Interleaf patent teaches (1: 15-27; 2: 19-25) an operating station that broadly reads on claim language. Claim language calls for graphically representable objects stored in memory, to be changed, deleted, appended and such functionality is similarly performed in Interleaf Patent's workstation.

Regarding **Ground of Rejection #4,** as obvious over the IBM reference in view of Interleaf and further in view of the Interleaf Patent and Lieberman. See Request pp. 10-13, and Exhibit I; received 07/16/2010. **Patent Owner** (p. 24) asserts "Lieberman is about automating user interaction by using scripts." Lieberman is related to "experiments in developing agent software that works with existing unmodified commercial applications and agents that work across multiple applications." **Patent Owner** argues (p. 24) that "Neither Samsung's request for reexamination nor the Office action explains how Lieberman relates to the field of the invention, interpretation of print data streams at a printer or print server. Lieberman has nothing to do with printing, print streams, printers, or print servers. There would have been no reason for one of ordinary skill in the art to refer to the Lieberman reference, or to use its teachings to solve the problems addressed in the '789 patent. (Birnbaum Dec. ¶ 96). Accordingly, the obviousness determination based on any combination involving the Lieberman reference is simply wrong - and is the result of improper hindsight analysis."

The Birnbaum Declaration (¶¶94-96) opines that Lieberman relates to agents that interact with multiple (existing) user applications. The field of the invention of the '789 patent is printer software generally, and in particular, interpretation or processing of print data streams at a printer or print server. Lieberman does not relate to printing, or to interpretation of print data streams at a printer or print server. It has nothing to do with the field of the invention, printing, print streams, printers, or print servers. There would have been no reason for one of ordinary skill to refer to
Lieberman, or to use its teachings to solve a problem relating to interpretation of print data streams
at a printer or print server.

Third Party Requester asserts (p. 16) that Patent Owner's arguments (that Lieberman is generally directed to scripts, and not specifically directed to using scripts in printers or print servers) is duplicative of their arguments addressed above, about prior art being in different fields of endeavor. (See comments discussing prior art, Requester Comments 09/06/2011, beginning at pg. 10.) The '789 patent also generally states that it is directed to computer programming: "The above described embodiments of the method according to the present invention can of course in each case also be implemented as a computer program product [in which] after the computer program has been loaded, the computer is caused by the program to carry out the method according to the invention described here." ('789 patent at 8:40-48.) One

skilled in the art would plainly be aware of the general and well-known uses of scripts, including the use of JavaScript -compiled or not.

The Jacobs Declaration (¶15) opines that each of the Interleaf reference, the Interleaf patent, and the Lieberman patent refers to scripts, and the common usage of scripts in various software scenarios. Scripting is a general-purpose, or multi-purpose software technique, and is not limited or restricted to one area of software technology. A person of ordinary skill in the art would be able to know and use scripts in many different situations. Consistent with this fact, the scripts described in the '789 patent, IBM, Interleaf, the Interleaf patent, and Lieberman all operate in a predictable manner. Using a type of script in one of these references in any other reference would function in its known, predictable manner. For example, the "Server-Side JavaScript Guide" shows examples of scripts also being used to manipulate files (Ex. R, received 09/06/2011, at pp. 170-177), send email messages (Ex. R, received 09/06/2011 at pp. 167-170), access databases (Ex. R, received 09/06/2011 at pp. 189-262), dynamically load functions and libraries (Ex. R, received 09/06/2011 at pp. 178-181), and other various functions. Exhibit S, received 09/06/2011, U.S. Pat. No. 6,678,705, has examples of scripts (in various languages, including Javascript) sending and receiving email (Ex. S, received 09/06/2011 at 3:8-4:30), retrieving information from Internet servers (Ex. S, received 09/06/2011 at 6:49-8:41), as well as interacting with various databases (Ex. S, 09/06/2011 at 8:50-11:23).

Examiner agrees with Requester. See Claim Chart Exhibit I, 07/16/2010, beginning at p. 46. Lieberman evidences Javascript as a common formal language for scripts. Lieberman provides evidence supporting Requesters position that specific language used in scripting is "merely a matter of obvious engineering choice" and thus *prima facie* obvious. (MPEP § 2141.04.V.(B)). Lieberman teaches that substituting Lisp (as taught in Interleaf) for Javascript is just substituting one known solution (Lisp) for another known solution (Javascript). Lieberman teaches (Abstract) a user interface agent for automating repetitive or semi-repetitive procedures (such as procedures written in script to affect printing objects) and suggests (p. 15) use of a scripting language. To relate Lieberman's teachings, broadly a method for the transformation of "print streams" may be integrated with script agents (Lieberman, p. 19, 5.2, a means through a scripting language to invoke the commands of the application). Lieberman (p. 17) states, "Agents can be seen as a way of supplying software that acts as the representative of the user's goals in the application environment. Agent software can provide glue between the applications, freeing the user from the complexity of dealing with the separate application environments (i.e., using script commands to deal with the complexity of outputting to various devices)."

The '789 patent suggests no unexpected or unpredicted benefit of using Java Script in this manner, and in fact merely refers to Java Script as one of several "example" scripting languages that could be used. ('789 patent, 6:10-11). The field of the endeavor of the '789 patent is not limited to printer software, generally printing, or to interpretation of print data streams at a printer or print server. The '789 Abstract clearly states, "A method for the transformation of digital print data streams...read in...analyzed by means of a parser...split up...stored...transformed

into a format for the control of an output device, preferably a printer..." A broad reading of the '789 patent includes transformations of data, involving script commands using objects / object-oriented format to more easily control external devices.

Examiner disagrees with Dr. Birnbaum's opinion (¶96) that the field of the invention of the '789 patent is printer software generally, and in particular, interpretation or processing of print data streams at a printer or print server. **Examiner** asserts that the invention is directed to transforming data streams (which are referred to as "digital print data streams"), using scripts to control output devices. Only preferably is the format of a data stream transformed into a format for the control of a printer output device. The '789 patent expressly states (5: 8-10) that the output device may broadly be all devices needed for document processing.

Contrary to Dr. Birnbaum's assertion (¶96), **Examiner** does not regard the '789 patent as recognizing a need to solve a problem relating to interpretation of print data streams at a printer or print server. Any need or problem known in the field of endeavor at the time of the invention and addressed by the patent [or application at issue] can provide a reason for combining the elements in the manner claimed. " *KSR International Co. v. Teleflex Inc.*, 550 U.S. ___, ___, 82 USPQ2d 1385, 1397 (2007). Thus a reference in a field different from that of Patent Owner's patent may be reasonably pertinent if it is one which, because of the matter with which it deals, logically would have commended itself to an inventor's attention in considering his or her invention as a whole. Notably the '789 patent (5: 3-6; 5: 38-54) teaches software objects

transformed, assigned at least one script – a single standardized programming language, to address the need to control external devices and reduce the effort on administration to a minimum. (emphasis added)

Examiner disagrees with Patent Owner's assertion that the rejections are based on improper hindsight reasoning. Any judgment on obviousness is in a sense necessarily a reconstruction based on hindsight reasoning, but so long as it takes into account only knowledge which was within the level of ordinary skill in the art at the time the claimed invention was made and does not include knowledge gleaned only from applicant's disclosure, such a reconstruction is proper." *In re McLaughlin* 443 F.2d 1392, 1395, 170 USPQ 209, 212 (CCPA 1971). Applicants may also argue that the combination of two or more references is "hindsight" because "express" motivation to combine the references is lacking. However, there is no requirement that an "express, written motivation to combine must appear in prior art references before a finding of obviousness." See *Ruiz v. A.B. Chance Co.*, 357 F.3d 1270, 1276, 69 USPQ2d 1686, 1690 (Fed. Cir. 2004). ** >

Examiner acknowledges Dr. Jacobs Exhibit R, received 09/06/2011, "Server-Side JavaScript Guide" shows examples of scripts also being used to manipulate files (Ex. R, received 09/06/2011 at pp. 170-177), send email messages (Ex. R, received 09/06/2011 at pp. 167-170), access databases (Ex. R, received 09/06/2011 at pp. 189-262), dynamically load functions and libraries (Ex. R, received 09/06/2011 at pp. 178-181), and other various functions. **Examiner** acknowledges Dr. Jacobs Exhibit S, received 09/06/2011, U.S. Pat. No. 6,678,705, has examples

of scripts (in various languages, including Javascript) sending and receiving email (Ex. S, received 09/06/2011 at 3:8-4:30), retrieving information from Internet servers (Ex. S, received 09/06/2011 at 6:49-8:41), as well as interacting with various databases (Ex. S, received 09/06/2011 at 8:50-11:23). The evidence supports the assertion that it was known to use scripts, including JavaScript to manipulate files.

Regarding **Ground of Rejection #5**, as anticipated by Interleaf. See Request pp. 10-14, and Exhibit J. **Patent Owner** asserts (p. 25-27) Interleaf is a "document creation component" (Interleaf, p. 75). The '789 Patent addresses an entirely different part of the workflow, namely, the printing stage. Consequently, there are at least four fundamental differences between Interleaf and the '789 Patent. Interleaf does not: • disclose a method for the transformation of print data streams; • read in an input print stream; • parse an input print data stream; or • produce an output print data stream based on a transformed input data stream. Patent Owner arguments (pp. 27-27) are duplicated in cites to Dr. Birnbaum Dec. at ¶¶99-107.

The Birnbaum Declaration (¶¶97-98) asserts the objective of the Interleaf system is to build an enhanced user interface (UI) and that Interleaf is a "document creation component" (Interleaf, p. 75). In contrast, the '789 patent deals with printing, which may include, for example, the output of user applications such as the Interleaf document creation system. The '789 patent addresses an entirely different part of the workflow, namely, the printing stage. That is, the patented technology takes over where the Interleaf system ends. **The Birnbaum Declaration** (¶¶99-107)

asserts at least four fundamental differences between Interleaf and the '789 patent: Interleaf does not disclose a method for the transformation of print data streams; read in an input print stream; parse an input print data stream; or produce an output print data stream based on a transformed input data stream. **Dr. Birnbaum** asserts:

(¶100), "The '789 patent claims a method, computer-readable medium, and computer signal for the transformation of digital print data streams. Statements at Interleaf, p. 76 and 81-84, refer to editing a working document in the user interface, not printing it."

(¶101), "The claims recite that an input print data stream is read in. Interleaf p. 84 ("If a document contains active objects within the document file stream.., these become activated when the document is opened programmatically or by the user for viewing, editing, or printing") refers to a document file stream, not a print data stream. The document file stream is merely what is read in by the Interleaf system when the Interleaf document is opened. It is not a print data stream, as recited in the claims."

(¶102), "Opening an Interleaf document is akin to opening a Word document...does not create a print data stream..."

(¶103), "...Interleaf may conceivably generate a print data stream as its output, but it does not read in a print data stream as its input..."

(¶104), "The claims recite that the input print data stream are analyzed by a parser for graphically representable objects. The Lisp interpreter operating on Interleaf does not operate on a

print data stream...it does not parse a print data stream."

(¶105), "...the Lisp interpreter (by definition) only interprets LISP scripts...it is incapable of interpreting print data streams...the parser...of the claims of the '789 patent...analyzes an input print data stream."

(¶106), "...Interleaf...may eventually create a print data stream it does not read that stream and analyze it using a parser and split it up into objects...Interleaf cannot transform objects and combine those transformed objects into an output print data stream."

(¶107), "Interleaf's "[o]pening the document for printing" does not: (1) read the print data stream; (2) parse it; (3) split it up into objects; (4) transform objects to control a printer; or (5) combine those transformed objects into an output data stream."

Dr. Birnbaum argues (¶108-114) the dependent claims:

(¶108), "For claims 2, 23, and 39, Interleaf's disclosure of a template document subclass with an 'open' method used for auto-localizing documents bears no relation to transforming a print data

stream."

(¶109), "Portions in Interleaf such as: (1) that active documents are defined as "structure documents.., in which the objects.., can be acted upon by, and can themselves act upon, other objects in the document or the outside world." and (2) the Intelligent Maintenance Aid (IMA) application example, which "make use of the system's printing and filtering capabilities" are irrelevant to claims 4, 25, and 41, which have to do with use of objects in connection with transformation of a print data stream..."

(¶110), "Regarding claims 5, 26, and 42, Interleaf's reference to LISP (e.g., pp. 77-78) does not suggest extracting the data at print time, that is, based on an input print data stream."

(¶111), "Regarding claims 6, 27, and 43, Interleaf's documents "not appear[ing] active to all users" further shows that Interleaf relates to user editing and handling of documents, not their printing."

(¶¶112-113), "Interleaf did not anticipate claims 7, 28, and 44 (or claims 11, 32, and 48), at least for the reason that it does not disclose every element of the independent claims from which these claims depend."

(¶114), "Regarding claims 12, 33, and 49, the cited features of Interleaf operate while the document is being edited or opened; they do not relate to manipulation of a print stream produced by an Interleaf document."

Third Party Requester (pp. 16-17) notes that the "PO basically makes one general argument about Interleaf--that the document creation described in Interleaf is "akin to opening a Word document," and is not about printing. (PO Remarks at 26.) Based on this argument, the PO then argues that Interleaf does not teach performing various steps on a "print data stream." This argument fails for several reasons, similar to those discussed above beginning at pg. 10, with reference to the fields of invention for the '789 patent and Interleaf Patent."

To summarize, the claims of the '789 patent recite processing steps that are performed in a computer, with the last step being outputting the processed print data stream to "an output

device,

preferably a printer." Interleaf is also directed to processing data in a computer, with the last step being providing the processed document file stream for (among other things) "printing. (Interleaf at 84.) Thus, Interleaf describes receiving a data stream for printing, regardless of it being called a "document file stream," and anticipates the above-listed claims of the '789 patent. Interleaf also states "The system is an excellent platform for the design, implementation, and delivery of active documents." (Interleaf at 75, emphasis added.)

Requester (p. 17) maintains that the above-listed claims are anticipated by Interleaf. In the event the Examiner finds that the document file stream taught in Interleaf is sufficiently distinct from the print data stream described in the '789 patent, the differences would be obvious in view of IBM. That is, it would be obvious for Interleaf to be modified to receive "print data streams" as taught in IBM, instead of "document file streams" as taught in Interleaf. Upon review of the teachings of Interleaf, a person of ordinary skill in the art would have recognized the interchangeability of using a document file stream shown in Interleaf vs. a print data stream. (Jacobs Dec.¶ 16-20; *see also,* MPEP 2183.B.) Furthermore, the document file stream in Interleaf performs the identical function of the print data stream in IBM, in substantially the same way, and produces substantially the same results. (Jacobs Dec.¶ 16-20; *see also,* MPEP 2183.A.) Further still, PostScript, a page description language used in IBM, was also commonly used for printing and display, so it would be a common choice of design for Interleaf to use PostScript ("PS") as its document file stream. (Jacobs Dec.¶ 16-20; *see also,* Ex. X, received 09/06/2011; wikipedia.org/ wiki/Adobe _Postscript; "As computer power grew, it became

possible to host the PS system in the computer rather than the printer. This led to the natural evolution of PS from a printing system to one that could also be used as the host's own graphics language.") The remaining elements of the claims have already been shown to be taught by Interleaf. Thus, in the alternative, claims 1-2, 4-7, 11 - 12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49 are obvious over Interleaf in view of IBM.

(See MPEP 2144.04.V.(B).)

The Jacobs Declaration asserts (addressing a possible combination of **Interleaf and IBM**) (¶¶16-18) that Interleaf describes transforming a "document file stream" into active documents. (Interleaf at 84.) The PO has made several arguments that a document file stream is not the same as a "print data stream," as that term is used in the '789 patent. I have been asked to consider this scenario, and whether it would be obvious for a person of ordinary skill in the art at the time of the filing of the '789 patent to use Interleaf technology with a print data stream (as shown in IBM) instead of a document file stream. The answer is yes, for several reasons. First, Interleaf and IBM both describe a computer system receiving a data stream, and performing the exact same operations as done in the '789 patent. ('789 patent at 8:40-48.) The received data stream performs the same function in each reference. Both references also refer to performing various operations on the transformed data stream, as discussed in the OA. Second, IBM lists "PostScript" as an example input print data stream. (IBM at 16.) It would have been obvious to use a PostScript data stream instead of a Lisp-oriented document stream--as Interleaf teaches--because the syntax of PostScript was based on and inspired by Lisp. *(See e.g.,* Ex. W, Thinking

in PostScript 1990, received 09/06/2011 at p. 1, "The first few chapters of this book help you put PostScript into perspective by

The Jacobs Declaration further opines (¶¶19-20) "...PostScript is useful for both displaying and printing documents. (Ex. U, "PostScript [is] used for displaying and printing," *see also,* Exhibits W and X; all received 09/06/2011.) In fact, I am aware of computer systems in the 1980s that received print data streams for displaying and editing documents. For example, the Sun Microsystem "NEWS" computer and the "NEXT" computer both used a version of PostScript display systems in the mid-1980's. *(See, e.g.,* Ex. X, wikipedia.org/wiki/AdobePostScript.) As noted above, the '789 patent lists PostScript as one example of an input print data stream. Thus, a person of ordinary skill in the art would have reason to use a print data stream in the Interleaf system.

Examiner broadly constructs the term "digital print data stream" as reading on a "digital" "data stream" where the stream may be sourced from a "document" and may or may not be output to a "printer." The '789 specification does not provide a narrow definition of "digital print data stream." Thus, Interleaf's "document file stream" anticipates the "digital print data stream" of the '789 patent. Interleaf does teach that following a transformation of the data stream, the stream may be output to a printer. Claim 1 transforms a "data stream," which is claimed as a "digital print data stream." The transformed data streams (transformed documents) are "output" to memory ('789, 3: 16-18), or "output" to a device, not necessarily a printer. It appears to

Examiner that a transformed document output (transformed data stream output) to memory or to a device such as an "archiving device" ('789, 5: 7) may not actually "print." The specification expressly states ('789, 5: 5-10) that the invention permits incorporation **all the devices needed in the widest sense for document processing**." (emphasis added) **Examiner** disagrees with Patent Owner statement that the '789 patent, in contrast to Interleaf, is directed to different part of the workflow, namely, the printing stage. At least one embodiment of the patent relates to the **transformation of objects, read in via input data stream and stored in memory, by assigning at least one script** that provides functionality to control external devices. (Id.) Similar to Interleaf, the '789 patent appears to source an "input digital print data stream" from a "document." See '789, 6: 61-65, "...therefore now also available for printing systems for the first time and here preferably forms the basic technology for **script applications in the area of** the present invention, and consequently **print and document management**, which is certainly uniquely and, as compared with established solutions, considerably more cost effective." (emphasis added) Transforming a document implicitly "reads in" data and "parses" code of the document.

Examiner disagrees with Dr. Birnbaum's assertion (¶100) that "[t]he '789 patent claims a method, computer-readable medium, and computer signal for the transformation of digital print data streams." The '789 patent claims a method, a system, a computer-readable medium, and a computer data signal embodied in a carrier wave and representing sequences of instructions which, when executed by a processor, cause the processor to perform a method, the method comprising..."

In response to Dr. Birnbaum's arguments directed towards the Lisp interpreter (Birnbaum Dec. ¶¶104-105), **Examiner** notes that Interleaf teachings do not preclude parsing / interpreting a data stream. See Request (Exhibit J Claim Chart, received 07/16/2010, pp. 4-5) states: "The input data stream is analyzed by means of a parser when it is "activated." The objects in the stream are analyzed by a Lisp interpreter: "Lisp was chosen as the I6 extension language because it is interpreted (providing a fast prototyping environment) and has run-time binding. Run-time binding is required for easy delivery of active documents." (Interleaf, p. 77) "Lisp is a good language for active document development due to features such as the interpreter (which allows nice development tools), functions as data (it is easy to store and manipulate programs as data on document objects), and run-time binding (which is important for the delivery of active document objects)." (Interleaf, p. 85) The "interpreter" is the parser as recited in the claim. The "run-time binding" of the "active document objects" by the Lisp interpreter shows that the data stream is analyzed by means of a parser as recited in the claim.

In response to Dr. Birnbaum's arguments directed towards parser functions (Birnbaum Dec. ¶¶105-107), **Examiner** notes that Interleaf teaches an "active load hook" that examines the code objects that are about to be activated (Lisp interpreter / parser analyzing the input document file stream). See more detailed explanation at Exhibit J - Claim Chart 07/16/2010, pp. 4-5. Document objects make up the document. (Interleaf, pp. 83-84) See Interleaf, pp. 81-82 for a teaching of a transformation to the object. Transformation capabilities are used to control publishing / printing and filtering capabilities. (Interleaf, p. 82)

In response to Patent Owner's remark and the Birnbaum Dec. at ¶107 and citing to Non Final Office Action 11/19/2010, p. 29 (Simply, "opening" a document for printing, plainly does not: (1) read the print data stream; (2) parse it; (3) split it up into objects; (4) transform objects to control a primer; or (5) combine those transformed objects into an output data stream), **Examiner** asserts that text was directed to the actual "printing" of the data stream. It did not address parsing, splitting, transforming, combining. Non Final Office Action 11/19/2010, p. 29: Interleaf teaches (p. 82)... In addition to having the presentation and editing functionality within the document, document system based applications can also take advantage of all the other publishing facilities that exist within the document system...**the system's printing** and filtering capabilities [transformed **document is received at output device / preferably a printer**]." The data stream that is sent to the system printing capability is analogous to an output print data stream. Opening the document for printing shows the system actually outputting the output print data stream."

Regarding Dr. Birnbaum's assertion at ¶110 (Interleaf's reference to LISP (e.g., pp. 77-78) does not suggest extracting the data at print time, that is, based on an input print data stream), **Examiner** asserts that the specification does not teach and the claim language does not recite the term "**print time**." The term "print time" is not germane to the issue. Regarding Dr. Birnbaum's assertion at ¶111 (Interleaf relates to user editing and handling of documents, not their printing), **Examiner** cites to '789 6: 58-65, "This intelligent technology...is...now also

available for printing systems...and here preferably form the basic technology for script applications in the area of the present invention, and consequently print and **document management**...considerably more cost-effective.", evidencing that the '789 patent also relates to editing (via scripts) and handling of documents. Analogously, "...Interleaf describes a system that receives a "**document file stream", transforms the stream** into "active" documents, and results in the "delivery of active documents." *(See, e.g.,* OA at 27-29; Interleaf at 75.) Interleaf further teaches that the "**active objects within the document file stream** ... become activated when the document is opened programmatically or by the user for viewing, editing, or **printing**." (Interleaf at 80.)" (emphasis added)

Examiner acknowledges the exhibits cited by Dr. Jacobs:

> **Exhibit** U received 09/06/2011 - "Computer Dictionary," Microsoft Press, Third Edition, 1997 (selected pages): PostScript n. A page-description language from Adobe Systems that offers flexible font capability and high-quality graphics. The most well-known page-description language, PostScript uses English-like commands to control page layout and to load and scale outline fonts. Adobe Systems is also responsible for Display PostScript, a graphics language for computer displays that gives users of both PostScript and Display PostScript absolute WYSIWYG (what-you-see-is-what-you-get), which is difficult when different methods are used for displaying and printing. *See also* outline font, page-description language.

Exhibit W received 09/06/2011 - Reid, "Thinking In PostScript" (1990) (selected pages): example excerpt from p. 2, "PostScript's most obvious language features are that it is an interpreted language... heavily oriented toward graphics and typography. These design choices were all made in order to make it useful as a device-independent page description language for imaging on raster devices... PostScript is heavily biased toward graphic imaging and typography because it was originally conceived as a mechanism to control raster imaging devices like laser printers and display screens."

Exhibit X received 09/06/2011 - en.wikipedia.org/wiki/Adobe_PostScript At p. 3: "PostScript went beyond the typical printer control language and was a complete programming language of its own. Many applications can transform a document into a PostScript program whose execution will result in the original document. This program can be sent to an interpreter in a printer, which results in a printed document, or to one inside another application, which will display the document on-screen. Since the document-program is the same regardless of its destination, it is called *device-independent*.

Examiner notes that in view of the collaborative nature of Wikipedia entries, citations to Wikipedia are given little evidentiary weight.

Examiner agrees with Third Party Requester and the Jacobs Declaration that Interleaf fairly anticipates claims 1-2, 4-7, 11-12, 22-23, 25-28, 32-33, 38-39, 41-44, and 48-49. See Request pp. 10-14 and Exhibit I. **Examiner** notes that Requester's suggestion for an alternate rejection is moot, as it was proposed as an alternative rejection based on Examiner accepting a premise.

Regarding **Ground of Rejection #6,** as obvious over the Interleaf reference in view of Lieberman. See Request pp. 13-15, and Exhibit J; 07/16/2010. **Patent Owner** asserts Lieberman does not involve printing whatsoever. A person of ordinary skill in the relevant art would have had no reason in 2000 to combine IBM or Interleaf with Lieberman for any reason. (Birnbaum Dec. ¶115). Suggesting that this combination would have been made is a classic exercise of hindsight analysis, using the patent at issue as template into which to place arbitrary prior work. The secondary indicia of non-obviousness also show why this conclusion is unsupported. Moreover, even if Lieberman and Interleaf were combined, for reasons explained above, the combination would not result in the claimed inventions (e.g., no parsing of an input print data stream). (Birnbaum Dec. ¶116).

The Birnbaum Declaration (¶¶115- 116) repeats Patent Owner's assertions (Lieberman does not involve printing...no reason to combine IBM or Interleaf with Lieberman...combined Lieberman and Interleaf would not result in the claimed inventions).

Third Party Requester (p. 17, in a combination response to Grounds of Rejection 6-9) asserts that "remaining arguments being made by the PO with respect to the claim rejections are repetitive of the ones made above, and for the sake of efficiency, the Requester will not duplicate the same comments further below. **Requester** adds, "...in the event that the Examiner finds that the "design, implementation, and delivery of active documents" described in Interleaf (at 75)

does not apply to print data streams, the Requester proposes that such would be obvious. (See discussion Third Party Comments, at pg. 16.)

Examiner agrees that arguments are repetitive and have been addressed. In response to Requester's comment regarding "print data streams," **Examiner** agrees that Interleaf (at 75) describes "data streams" that are sufficiently analogous to the '789 specification. See discussion above regarding the broad interpretation of the term "input print data stream." See Non Final Office Action 11/19/2010, p. 27-29: Interleaf transforms objects in the stream. **Examiner** notes that Non Final Office Action (11/19/2010, p. 29) states that Interleaf (p. 82) teaches a **data stream [digital input print data stream] sent to the system printing**. (emphasis added) See **Examiner** "hindsight" rebuttal at ACP, p. 140.

Regarding **Grounds of Rejection #7,** as obvious over the Interleaf reference in view of the Interleaf Patent. See Request p. 15, and Exhibit K; received 07/16/2010. **Patent Owner** asserts (pp. 27-28): The Interleaf Patent does not rectify the deficiencies of the Interleaf reference disclosed above...does not relate to transforming a print data stream. **Patent Owner** notes that Examiner referred to the following in the Interleaf Patent (Col. 3, lines 37-39) (to purportedly find a script executed in the cases defined in the script): "any data object within the hierarchy may be associated with a method which, upon the occurrence of a certain event, will be implemented." However, this must be read in context. The paragraph reads (emphasis added): [A]ny data object within the hierarchy may be associated with a method which, upon the

occurrence of a certain event, will be implemented. For example, referring to FIG. 5, memorandum 32 is associated with an open method 64. When a user, via user interface 30, indicates that he desires to view memorandum 32, the open method 64 is implemented to open the memorandum file 31 and prepare image data representative of how the document would appear if printed. The image data is supplied to display device 24 which generates an image of the document as shown in FIG. 3. The occurrences upon which methods are implemented are in the context of user interactions with the document, e.g., user opening a document, clicking on an image, etc. They are not in a print data stream, nor are they implemented in the context of printing a document, as claimed in the '789 Patent. (Birnbaum Dec. ¶ 118).

The Birnbaum Declaration (¶¶117-129) asserts...the Interleaf Patent, like Interleaf, does not relate to transforming a print data stream. **Dr. Birnbaum** repeats the exemplary citation presented by Patent Owner, noting "...it is clear that the occurrences upon which methods are implemented are in the context of user interactions with the document, e.g., user opening a document, clicking on an image. They are not in a print data stream, nor are they implemented in the context of printing a document, as claimed in the '789 patent.

Regarding claims 2, 23, and 39, neither Interleaf nor the Interleaf Patent bears any relation to transforming a print data stream.

Regarding claims 4, 25, and 41, as with the example provided by a "phone directory that can call the person who's [sic] name you've selected; if the line is busy, it puts you into electronic mail," the cited portions of the Interleaf documents relate to a user-document interface, not a printing function.

Regarding claims 5, 26, and 42 and claims 6, 27, and 43, the cited portions of Interleaf and the Patent do not relate to use in connection with an input print data stream. For example, one application involves retrieving stock information and displaying it in text and charts. However, this capability is discussed with respect to the limited scope of document creation and editing. Once the document is sent to the printer, that functionality is lost. Nothing in the references suggests obtaining such information during the printing process, i.e., after receiving an input print
data stream.

Regarding claims 7, 28, and 44, the portion of the Interleaf Patent that discloses a "customer receipt that automatically runs a 'frame grabber' and incorporates a photographic image of the purchaser" does not transpire based on an input print data stream, but during creation of a document.

Regarding claims 8, 29, and 45, the portions of the Interleaf Patent, including an "urgent memo that integrates with voice annotation software so when it mails itself to someone, it announces its presence audibly" and a "document created from a database which updates the database if you make any changes in the imported document" have no application in a print application.

Regarding claims 9, 30, and 46, the Interleaf Patent's cited disclosure has no application to a transformation of a print data stream, as recited in the claims.

Regarding claims 10, 31, and 47, the Interleaf Patent's example of a "stock-market information center that retrieves stock information and displays it in text and charts that can be automatically

distributed or incorporated in other documents" does not suggest obtaining such information after receiving an input print data stream.

Regarding claims 11, 32, and 48, the cited Interleaf Patent at col. 6, lines 6-16, and col. 3, lines 37-39, does not disclose or otherwise relate to transformation of a print data stream.

Regarding claims 12-14, 33-35, and 49-51, the cited portion of the Interleaf Patent has no application to a transformation of a print data stream, as recited in the '789 patent claims. In addition, the examples, e.g., a document that includes "Draft" before an announcement date, a reminder that appears on a screen based on a document due date, relate to the generation of a document, not an operation performed on a print stream based on a document. They are not relevant to the claims.

Regarding claims 16, 37, and 53, the various editing functions disclosed in the Interleaf Patent are performed before generating an input print data stream. Nowhere does the Patent disclose or render obvious that these functions are performed after receiving an input print data stream, as required by the claims.

Regarding claim 18, the cited functions of the Interleaf Patent occur before the objects are output in the output print data stream. The Patent does not disclose performing them after receiving a print data stream.

Examiner asserts that many of the above arguments are improperly based on a narrow interpretation of claim language. Patent Owner and Dr. Birnbaum repeat arguments that the transformation within Interleaf is "document generating / creating / editing," not occurring on an

"print data stream." **Examiner** disagrees. The transformation on the data stream taught in Interleaf is analogous to the '789 "modifications" that take place on a data stream at the "operating station with display means." See '789, FIG. 1, #8. Only "preferably" are the objects sent to an output printer device.

Regarding **Ground of Rejection #8**, as obvious over the Interleaf reference in view of the Interleaf Patent and further in view of Lieberman. Office action, p. 38), **Patent Owner** asserts (p. 28): ...neither of the Interleaf Documents, nor Lieberman involve printing whatsoever. Combining three non-printing references, as a matter of common sense, cannot create a printing invention. Further, a person of ordinary skill in the relevant art would have had no reason in 2000 to combine the Interleaf Documents with Lieberman for any reason. Suggesting that this combination would have been made is a classic exercise of hindsight analysis, using the patent at issue as template into which to place arbitrary prior work. (Birnbaum Dec. ¶130). In the end, the combination is not what is claimed.

Dr. Birnbaum (¶130) repeats the opinion regarding Ground of Rejection No. 4: Lieberman does not involve printing. A person of ordinary skill in the relevant art would have had no reason to combine IBM or the Interleaf references with Lieberman for any reason.

Examiner repeats that Interleaf is directed towards transforming a data stream to control publishing (printing). See Non Final Office Action (11/19/2010, pp. 27-30) and Interleaf, pp. 76

& 80-84. Interleaf/Interleaf Patent fail to disclose Java Script objects used as the formal language for the scripts. Lieberman (p. 17) is introduced into the combination to evidence Java script as a common formal language for scripts. Lieberman teaches that substituting Lisp (as taught in Interleaf) for Java script is just substituting one known solution (Lisp) for another known solution (Java script). 2) A person of ordinary skill in the art would be motivated to combine Interleaf and Lieberman because they both address the same field, namely scripting interfaces for object systems. This overlap in the field of use would motivate a person of ordinary skill in the art to combine Interleaf and Lieberman. See Non Final Office Action (11/19/2010, pp. 38-40). Regarding Dr. Birnbaum's statement at ¶130, **Examiner** notes that this ground or rejection does not involve IBM.

Regarding **<u>Ground of Rejection #9,</u>** as obvious over the Interleaf Patent in view of IBM. Office action, p. 40. **Patent Owner** asserts (p. 28): ...remarks pertaining to the impermissible combination of the Interleaf Documents and the IBM Reference are fully applicable here. "...there is no overlap between the Interleaf Patent and the IBM Reference...the print data stream produced by the Interleaf Patent as an output is handed off to the IBM Reference as an input. (Birnbaum Dec. ¶ 132). It makes no sense to "combine" the references in the manner suggested by Samsung, or to apply processes of the IBM Reference (print formatting and printer functionality) to those of the Interleaf Patent (document creation and editing). Furthermore, reference in the Office action to various types of print data streams is irrelevant, because these are simply converted to AFP data streams for processing. (Birnbaum Dec. ¶ 133).

The Birnbaum Declaration (¶¶131-133) repeats Patent Owner assertions and opines: The above remarks pertaining to the combination of the Interleaf Patent and IBM are incorporated herein by reference, and are not repeated...there is no overlap between the Interleaf Patent and IBM. The print data stream produced by the Interleaf Patent as an output is handed off to IBM as an input...this combination would not result in any claimed invention.

Examiner asserts that the Interleaf Patent and IBM are properly combined prior art references. The Interleaf Patent describes an "electronic document processing system" or EDPS that modifies documents for printing. Interleaf Patent at 1:8-11; 2:10-11. Fig. 5 of the patent, shows an example document in the form of a memorandum 32 having multiple objects, including a "graphic object 48." Interleaf Patent, 3:4. The Interleaf Patent describes providing scripts to the objects to trigger events during presentation or printing. A list of example events is provided in columns 6-9 of the patent. Interleaf Patent creates electronic documents for "printing." IBM describes the Advanced Function Presentation and printing system (AFP). **Examiner** asserts that the proposed modification does not cause a print data stream produced by the Interleaf Patent as an output to be handed off to IBM as an input. The Interleaf Patent teaches a method for transformation of digital print data streams (graphic objects and object oriented format) using scripting. IBM teaches reading in (parses) and transforming an input data stream. Similarly, IBM teaches DDS commands (scripting). The IBM teachings that parse, transform and script, combined with the object / scripting transformation teachings of Interleaf would operate in a manner similar to '789 FIG. 1. Transformation of a data stream occurs similarly to the claimed

'789 invention within the "operating station, #8." Notably **Examiner** does not narrowly interpret the term "input print data stream," as the data stream is only preferably printed.

Patent Owner further discusses (pp. 29-33) the dependent claims.

Examiner finds that arguments are repetitive where individual issues are covered above, citing to paragraphs of the Birnbaum Declaration, which also were acknowledged above. It is clear that Patent Owner argues for a narrow interpretation of objects, a print data stream, scripts, scripts assigned to objects, objects stored in object-oriented format. As asserted above, reexamination takes the broadest reasonable interpretation. **Examiner** finds no support for "print time" in the specification. Arguments related to operations that are performed after receiving an input print data stream are unclear. When does a data stream become an input print data stream? Some of the data streams of the '789 patent, even though they are called an "input print data stream" are not actually output to a printer. Regarding the argument for claim 20 (Patent Owner Comments, p. 33), the AS/400 connected to a printer would likely operate similar to the '789 FIG. 1, which shows #3, a PC computer connected to #9, an output device that <u>may</u> be a printer. Notably FIG. 1 is the only figure of the entire patent.

USPTO classifications overlap subject matter. The '789 patent is classified under 101/484 (and related 101 subclasses) and also under 400/61 (and related subclasses). Classification 101 relates to machines which print and sort, with subclass /484 addressing a process step performed in response to a sensed changing or abnormal condition of circumstance. Classification 400 relates

to typewriting machines, with subclass /61 addressing control of format and selection of type-face by programmed control system. The '789 patent also acknowledges a search in the 358 classification, which reads on one embodiment of the claimed system, a facsimile output device. While not noted on the face of the '789 patent, the subject matter also fairly reads on the 717 classification (Software Development, Installation, and Management; 717/112 editing, syntax based; 717/115 script; 717/143 parsing, syntax analysis, and semantic analysis).

In view of the foregoing, regarding the **Declaration of David Birnbaum**, based on consideration of the entire record by a preponderance of the evidence, when all of the evidence is considered, opinions and evidence presented fail to outweigh the evidence presented by Third Party Requester. The Birnbaum declaration, received 03/30/2011, is insufficient to overcome the rejections under 35 USC 102(b) and 103(a) for the reasons detailed in the above claim rejections, and responses to arguments.

In view of the foregoing, regarding the **Declaration of Roland Widuch**, based on consideration of the entire record by a preponderance of the evidence, weighted by the relevance, when all of the evidence is considered, the totality of the rebuttal evidence claiming commercial success fails to outweigh the evidence presented by Third Party Requester. The Widuch Declaration, filed 01/10/2011, is insufficient to overcome the rejections under 35 USC 102(b) and 103(a) for the reasons detailed in the claim rejections, and responses to arguments.

In view of the foregoing, regarding the **Declaration of Christoph Picht**, based on consideration of the entire record by a preponderance of the evidence, weighted by the relevance, when all of the evidence is considered, the totality of the evidence fails to outweigh the evidence presented by Third Party Requester. The Picht Declaration, received 03/30/2011, is insufficient to overcome the rejections under 35 USC 102(b) and 103(a) for the reasons detailed in the claim rejections, and responses to arguments.

In view of the foregoing, regarding the **Declaration of Paul Jacobs** under 37 CFR 1.132, based on consideration of the entire record by a preponderance of the evidence, when all of the arguments and evidence are considered, the argument to maintain rejections outweighs the arguments and evidence presented by Patent Owner and declarants Widuch, Picht, and Birnbaum. The Jacobs declaration under 37 CFR 1.132 in combination with Third Party Requester Comments, received 09/06/2011, are found persuasive and sufficient to maintain the rejections under 35 USC 102(b) and 103(a) for the reasons detailed in the following claim rejections, and responses to arguments.

Examiner has reviewed the original disclosure, weighed any evidence relied upon in establishing the prima facie showing, any claim amendments, and any new reasoning or evidence provided by the Patent Owner and Third Party Requester relative to asserted specific and substantial credible utility of the '789 patent. **Examiner** has fully considered and responded (within the Response to Arguments section) to each substantive element of Patent Owner Remarks, the Third Party Requester Comments and the Declarations. All Declarations and

supporting exhibits have been considered and entered into the record. Patentability is determined on the totality of the record, by a preponderance of evidence with due consideration to persuasiveness of argument. The evidence is deemed insufficient to rebut the prima facie case of obviousness. **Examiner** has considered the Patent Owner's submissions under "secondary considerations." The record has established a strong case of obviousness with respect to the various rejections based on combinations of IBM, Interleaf, Interleaf patent and Lieberman. **Examiner** maintains that the obvious rejections provide an articulated reasoning with rational underpinning to support the legal conclusion of obviousness. Examiner maintains the rejections of claims 1-53 and adopts the rejections proposed for newly added claims 54-82. Examiner maintains that claims are given the broadest reasonable interpretation, consistent with the Specification (Trans Tex. Holdings).

Information Disclosure Statement

IDS received 04/26/2011 has been entered into the record. 37 CFR 1.555(a) states, in part: Any information disclosure statement must be filed with the items listed in § 1.98(a) as applied to individuals associated with the patent owner in a reexamination proceeding, and should be filed within two months of the date of the order for reexamination, or as soon thereafter as possible. "With respect to the **Information Disclosure Statement** (PTO/SB/08A and 08B or its equivalent) considered with this action, the information cited has been considered as described in the MPEP. Note that MPEP 2256 and 2656 indicate that degree of consideration to be given to such information will be normally limited by the degree to which the party filing the information citation has explained the content and relevance of the information. A concise explanation of the

relevance, as it is presently understood by the individual designated in § 1.56(c) most knowledgeable about the content of the information, of each patent, publication, or other information listed that is not in the English language may be either separate from applicant's specification or incorporated therein.

Conclusion

This is an **ACTION CLOSING PROSECUTION (ACP)**; see MPEP § 2671.02.

(1) Pursuant to 37 CFR 1.951 (a), the patent owner may once file written comments limited to the issues raised in the reexamination proceeding and/or present a proposed amendment to the claims which amendment will be subject to the criteria of 37 CFR 1.116 as to whether it shall be entered and considered. Such comments and/or proposed amendments must be filed within a time period of 30 days or one month (whichever is longer) from the mailing date of this action. Where the patent owner files such comments and/or a proposed amendment, the third party requester may once file comments under 37 CFR 1.951 (b) responding to the patent owner's submission within 30 days from the date of service of the patent owner's submission on the third party requester.

(2) If the patent owner does not timely file comments and/or a proposed amendment pursuant to 37 CFR 1.951 (a), then the third party requester is precluded from filing comments under 37 CFR 1.951(b).

(3) Appeal cannot be taken from this action, since it is not a final Office action.

Extensions of time under 37 CFR 1.136(a) will not be permitted in *inter partes* reexamination proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to the patent owner in a reexamination proceeding. Additionally, 35 U.S.C. 314(c) requires that inter partes reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.937). Patent owner extensions of time in inter partes reexamination proceedings are provided for in 37 CFR 1.956. Extensions of time are not available for third party requester comments, because a comment period of 30 days from service of patent owner's response is set by statute. 35 U.S.C. 314(b)(3).

Any paper filed with the USPTO, i.e., any submission made, by either the Patent Owner or the Third Party requester must be served on every other party in the reexamination proceeding, including any other Third Party requester that is part of the proceeding due to merger of the reexamination proceedings. As proof of service, the party submitting the paper to the Office must attach a Certificate of Service to the paper, which sets forth the name and address of the party served and the method of service. Papers filed without the required Certificate of Service may be denied consideration. 37 CFR 1.903; MPEP 2666.06.

The Patent Owner is reminded that any proposed amendment to the specification and/or claims in this reexamination proceeding must comply with 37 CFR 1.530(d)-(j), must be formally presented pursuant to 37 CFR 1.52(a) and (b), and must contain any fees required by 37 CFR 1.20(c). Amendments in an *inter partes* reexamination proceeding are made in the same manner

that amendments in an *ex parte* reexamination are made. MPEP 2666.01. See MPEP 2250 for guidance as to the manner of making amendments in a reexamination proceeding.

The Patent Owner is reminded of the continuing responsibility under 37 CFR 1.985(a), to apprise the Office of any litigation activity, or other prior or concurrent proceeding, involving the instant Patent Under Reexamination or any related patent throughout the course of this reexamination proceeding. The Third Party requester is also reminded of the ability to similarly inform the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP §§ 2686 and 2286.04.

All correspondence relating to this *Inter Partes* reexamination proceeding should be directed:
By EFS: Registered users may submit via the electronic filing system EFS-Web, at
https://efs.uspto.gov/efile/myportal/efs-registered

By Mail to: Mail Stop *Inter Partes* Reexam
Attn: Central Reexamination Unit
Commissioner for Patents United States Patent & Trademark Office
P.O. Box 1450
Alexandria, Virginia 22313-1450

By FAX to: (571) 273-9900

Central Reexamination Unit

By hand: Customer Service Window

Attn: Central Reexamination Unit

Randolph Building, Lobby Level

401 Dulany Street

Alexandria, VA 22314

Any inquiry concerning this communication or earlier communications from the examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

/Mary Steelman/

Primary Examiner

Central Reexamination Unit - Art Unit 3992

(571) 272-3704

Conferees: /EBK/

ALEXANDER J. KOSOWSKI
Supervisory Patent Reexamination Specialist
CRU -- Art Unit 3992

AJK

> AS/400 now provides a wide range of printing support including AFP printers of various technologies, speeds, and capabilities. AS/400 has also extended print support to the network, enabling it to serve clients and printers on the network.

IBM at 11, emphasis added. Thus, the AS/400 has an attached printer already, and the IBM reference teaches how the AS/400 can also provide a print data stream to printers on the network.

This is further shown in Fig. 178 at page 289 of IBM, reproduced and annotated below. The AS/400 with its attached line printer combined are "a printer" (highlighted in the figure with a dotted box) that outputs a "print data stream" to a printer (one of the two networked printers on the right side of the figure).[1] Thus, IBM reads on the claims.

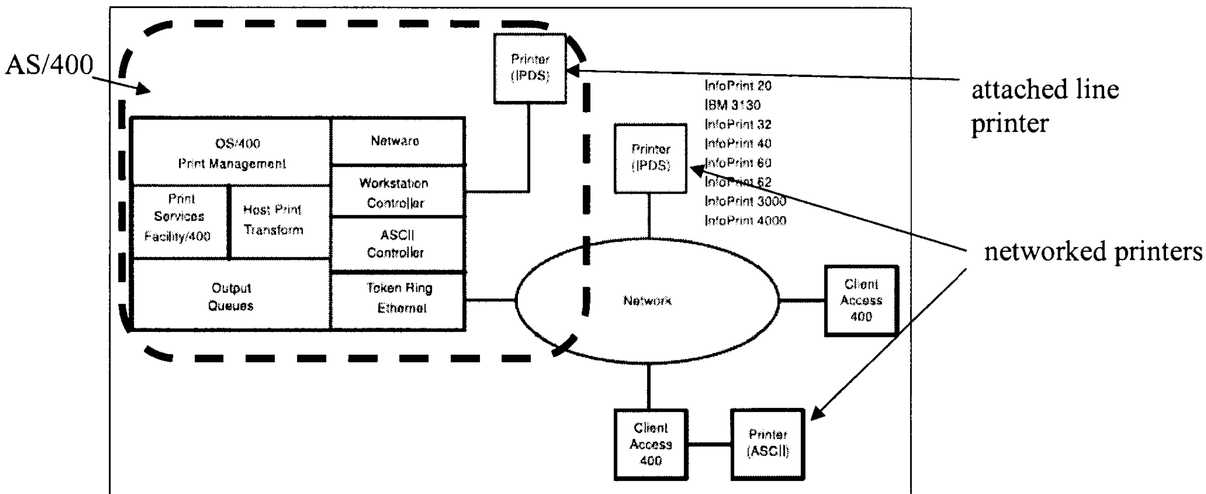


Figure 178. Printing LAN applications on workgroup printers

IBM, p. 268

Likewise, the Interleaf patent is directed towards "a Document Processing System for creating, editing, printing and presenting active electronic documents." 1:8-11. Thus, the "system" of the Interleaf patent is for "printing," and can therefore be considered a "printer."

Finally, the background of the '789 patent shows that intelligent printers (*i.e.*, printers with processing power) were not new and that it was known in the art to put processing functionality on printers. *See, e.g.*, 1:17-21; 2:44-48. The purported novelty of this patent was not to put the processing on a printer as opposed to a computer; it was the processing method itself. *See* 2:62-3:4.

---

[1] It is noted that the claims do not recite that the system and first printer is contained in a single, common chassis.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
95/001,398	07/16/2010	6684789	P-74637-US	5488

49443 7590 07/30/2012
Pearl Cohen Zedek Latzer, LLP
1500 Broadway
12th Floor
New York, NY 10036

EXAMINER	
STEELMAN, MARY J	

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
07/30/2012	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

PTOL-90A (Rev. 04/07)

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patents and Trademark Office
P.O.Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

DO NOT USE IN PALM PRINTER

THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS
HAYNES AND BOONE, LLP
IP SECTION
2323 VICTORY AVENUE, SUITE 700
DALLAS, TX 75219

Date:
MAILED
JUL 3 0 2012
CENTRAL REEXAMINATION UNIT

Transmittal of Communication to Third Party Requester
Inter Partes Reexamination

REEXAMINATION CONTROL NO. : 95001398
PATENT NO. : 6684789
TECHNOLOGY CENTER : 3999
ART UNIT : 3992

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified Reexamination proceeding. 37 CFR 1.903.

Prior to the filing of a Notice of Appeal, each time the patent owner responds to this communication, the third party requester of the inter partes reexamination may once file written comments within a period of 30 days from the date of service of the patent owner's response. This 30-day time period is statutory (35 U.S.C. 314(b)(2)), and, as such, it cannot be extended. See also 37 CFR 1.947.

If an ex parte reexamination has been merged with the inter partes reexamination, no responsive submission by any ex parte third party requester is permitted.

All correspondence relating to this inter partes reexamination proceeding should be directed to the Central Reexamination Unit at the mail, FAX, or hand-carry addresses given at the end of the communication enclosed with this transmittal.

PTOL-2070(Rev.07-04)

Transmittal of Communication to Third Party Requester Inter Partes Reexamination	Control No. 95/001,398	Patent Under Reexamination 6684789
	Examiner MARY STEELMAN	Art Unit 3992

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above-identified reexamination proceeding. 37 CFR 1.903.

Prior to the filing of a Notice of Appeal, each time the patent owner responds to this communication, the third party requester of the *inter partes* reexamination may once file written comments within a period of 30 days from the date of service of the patent owner's response. This 30-day time period is statutory (35 U.S.C. 314(b)(2)), and, as such, it cannot be extended. See also 37 CFR 1.947.

If an *ex parte* reexamination has been merged with the *inter partes* reexamination, no responsive submission by any *ex parte* third party requester is permitted.

All correspondence relating to this inter partes reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of the communication enclosed with this transmittal.

Right of Appeal Notice (37 CFR 1.953)	Control No. 95/001,398	Patent Under Reexamination 6684789
	Examiner MARY STEELMAN	Art Unit 3992

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address. --

Responsive to the communication(s) filed by:
Patent Owner on 09 May, 2012
Third Party(ies) on 07 June, 2012

Patent owner and/or third party requester(s) may file a notice of appeal with respect to any adverse decision with payment of the fee set forth in 37 CFR 41.20(b)(1) within **one-month or thirty-days (whichever is longer)**. See MPEP 2671. In addition, a party may file a notice of **cross** appeal and pay the 37 CFR 41.20(b)(1) fee **within fourteen days of service** of an opposing party's timely filed notice of appeal. See MPEP 2672.

All correspondence relating to this inter partes reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of this Office action.

If no party timely files a notice of appeal, prosecution on the merits of this reexamination proceeding will be concluded, and the Director of the USPTO will proceed to issue and publish a certificate under 37 CFR 1.997 in accordance with this Office action.

The proposed amendment filed 09 May, 2012 ☒ will be entered ☐ will not be entered*

*Reasons for non-entry are given in the body of this notice.

1a. ☒ Claims 1-64,66-72 and 74-82 are subject to reexamination.
1b. ☐ Claims _____ are not subject to reexamination.
2. ☒ Claims 65 and 73 have been cancelled.
3. ☐ Claims _____ are confirmed. [Unamended patent claims].
4. ☐ Claims _____ are patentable. [Amended or new claims].
5. ☒ Claims 1-64,66-72 and 74-82 are rejected.
6. ☐ Claims _____ are objected to.
7. ☐ The drawings filed on _____ ☐ are acceptable. ☐ are not acceptable.
8. ☐ The drawing correction request filed on _____ is ☐ approved. ☐ disapproved.
9. ☐ Acknowledgment is made of the claim for priority under 35 U.S.C. 119 (a)-(d) or (f). The certified copy has:
☐ been received. ☐ not been received. ☐ been filed in Application/Control No. _____.
10. ☐ Other _____

Attachments

1. ☐ Notice of References Cited by Examiner, PTO-892
2. ☐ Information Disclosure Citation, PTO/SB/08
3. ☐ _____

RIGHT OF APPEAL NOTICE

This office action addresses the reexamination of USPN 6,684,789 to Krautter. Patent Owner Comments (05/09/2012, incorporated by reference) and Third Party Requester Comments (06/07/2012, incorporated by reference) after ACP have been entered into the record and given full consideration, as noted below.

Per Action Closing Prosecution (04/09/2012), claims 1-82 are rejected. No claims are confirmed. Claim 9 was previously amended. Per Patent Owner's request (05/09/2012, claims 65 and 73 are cancelled. Per Patent Owner's request (05/09/2012), the language of New Claims 56, 57, 59, 62, 66, 68, and 74 has been revised.

Response to Arguments

I. Rejections Under 35 USC 112

A. Regarding the 35 USC 112 second paragraph rejection of **claims 56, 62, 66, and 74** (omitting essential steps, and in particular, an enabling disclosure showing how to print without a print driver), **Patent Owner** asserts (pp. 3-5):

Patent Owner cites to the specification ('789, 5: 38-59; 6: 38-51) for support and asserts, "It is clear that the ability to print without a printer driver is not an independent step, but a result of using the method and system of the invention of the '789 patent.

For example, claim 56 depends from claim 54. Claim 54 recites a system for the transformation of digital print data streams having a data processing unit programmed... Claim 56 (as revised herein) recites the system of claim 54, characterized in that the data processing unit is further programmed to transform the graphically representable objects into a format for the control of a printer and combine the objects into an output print data stream without a printer driver.

By splitting up the input print data stream into graphically representable objects, storing the graphically representable objects in an object-oriented format, transforming the graphically representable objects into a format for the control of a printer, combining the objects into an output print data stream, and outputting the combined output print data stream to said printer, printing is enabled without the use of a printer driver. Claims 62, 66, and 74 have been similarly revised.

Examiner acknowledges that Patent Owner has modified claim language and provided additional citations in support of claim limitations. Effectively printer driver software is programmed into a data processing unit. **Examiner** hereby withdraws the 35 U.S.C. 112, second paragraph rejection of claims 56, 62, 66, and 74 (as being incomplete for omitting essential steps, such omission amounting to a gap between the steps).

B. Patent Owner points out (p. 5) that **claims 57 and 59** have been revised to overcome a lack of antecedent basis.

Examiner presumes a typo (Patent Owner Remarks 05/09/2012, p. 5) and that Patent Owner is referring to claims 57 and 58. Claim 57 has been properly amended to overcome a lack of antecedent basis, and thus claim 58 is proper. The 112 second paragraph rejections of claims 57 and 58, based on lack of antecedent basis, are withdrawn.

C. Regarding the 35 USC 112 second paragraph rejections of **claims 59 and 64, Patent Owner** explains (pp. 5-7) how a printer can output a print data stream. **Patent Owner** cites to '789, 2: 55-61; 2: 62-3: 4; 5: 21-31; 5: 60 – 6: 5; 5: 32-37. "The specification discloses at length transforming digital print data streams so as to allow for additional functionality..." "Thus, a print data stream of the invention may be processed by various output devices."

Patent Owner asserts (p. 7), "Thus, in addition to printing an output print data stream, a printer server or a printer may send objects in the output print data stream to another device for further processing. Moreover, the software may be installed in a printer server or printer, such that a printer can perform these functions to the same extent as a printer server:

The system according to the invention can also be integrated into a printer or else a printer server. (col. 8, lines 24-25)

In addition, the functionality of a printer driving another printer is specifically described in the White Paper attached as Exhibit F to the Widuch Declaration (p. 2), discussed in further detail below. In particular, the White Paper states that JScribe applications allow printers to control connected peripheral devices, including RFID printers, label printers, and slave printers. Thus, it

is clear throughout the disclosure that by the use of the system and method of the invention, printers are made capable of outputting a print data stream, for example, to another output device in the network.

Third Party Requester rebuts (pp. 1-2) Patent Owner logic (PO Remarks at 7) arguing the printer can send a print data stream to *another* printer is not described in any of the quotes listed by the inventor, nor is it described in the '789 patent specification. Instead, the patent specification describes the printer doing operations, such as operating envelope and paper-folding systems (5:3-10), sending and receiving emails (5:38-41), and accessing the internet for information (6:31-38) in response to objects and scripts in the print data stream. The only citation provided by the PO that refers to a printer driving another printer is a White Paper. (PO Remarks at 7) But nowhere does *the patent specification* teach a printer outputting a print data stream to another printer.

Examiner agrees with Requester that the '789 specification fails to teach "...wherein said printer is adapted...to output said combined output print data stream..." as found in claims 59 and 64. Support for claim limitations may not be found in related non patent literature outside the specification. See ACP, pp. 79-90 regarding Secondary Considerations of Non-Obviousness. See ACP, p. 84 that identifies Exhibit F (attached to the Widuch Declaration, 01/10/2011) as undated. Expert opinion weight is influenced by the underlying basis of facts. **Examiner** maintains the 35 U.S.C. 112 second paragraph rejection of claims 59 and 64 (as unclear as to

how a printer can output a print data stream), and claims 60-63 and 65-67 due to dependence on claims 59 and 64.

II. Rejections Under 35 U.S.C. 314(a)

A. Regarding the 35 USC 314(a) rejections of **claims 59 and 68**, **Patent Owner** notes (p. 8) that claim language has been revised to include the terms "data processing unit" and "communications interface."

In view of amended claim language (claims 59 and 68), **Examiner** hereby withdraws the prior 35 USC 314(a) rejection (claims 59, 60-63, 68, and 69-71).

B. Regarding the 35 USC 314(a) rejections of **claims 64 and 72**, **Patent Owner** notes (pp. 8-9) that claims 64 and 72 have been revised to include subject matter previously recited in claim 18.

Examiner notes that original independent claims 1, 17, 22, and 38 recite: "A method..."; "A system..."; "A computer readable medium...the method comprising..."; and "A computer data signal embodied in a carrier wave...the method comprising..." There is no original claim language that recites the scope "A printing system." Thus, the proposed new claims enlarge the scope of the patented claims. Newly proposed claims 64 and 72 lack support in original claims. Dependent claims 65-67 and 73-75 fail to rectify the omission and are therefore rejected under the same reasoning. **Examiner** maintains the 35 U.S.C. 314(a) rejections of claims 64-67 and 72-75.

III. Rejections Under 35 U.S.C. Sections 102 and 103

A. Regarding the rejections of **claims 20, 59-63, and 64-67**, **Patent Owner** asserts (pp. 9-10) that none of the references disclose or render obvious these claims, and objective evidence of non-obviousness, including a clear nexus, is particularly relevant to these claims:

1. Patent Owner asserts (PO Remarks 05/09/2012, pp. 10-12) that the prior art does not disclose or render obvious claims covering a printer. Regarding Ground No. 1, the ACP states that the IBM reference discloses using the AS/400 to communicate with printers, and that the printer devices have data processing units with memory (ACP, p. 29). Similarly, Requester's Exhibit G merely states that IBM teaches a printer (citing to IBM, p. 1), but fails to show that the printers disclosed in the IBM reference have the system or functionality recited in claims 17 and 1 (Exhibit G, pp. 24-25). That is, even according to the ACP, it is only the AS/400 that performs the print data stream processing, and sends the output stream to the printer. For example, the DDS, which the Examiner has considered to be a script, runs on the AS/400, not on a printer. In contrast, claim 20 recites a printer that itself has "a system for the transformation of digital print data streams" as claim in claim 17. The ACP merely purports to show that the AS/400 is a system for the transformation of digital print data streams, and that it communicates with a printer, not that the IBM reference discloses a printer capable of transforming digital print data streams as claim in claims 17 and 1. Nor has the Examiner shown that it would have been obvious to incorporate the purported functionality of the AS/400 into a printer.

Likewise, under Grounds No. 2 and 3, the ACP states that "IBM teaches a printer / printer server characterized in that it has a system for the transformation of digital print data streams." (ACP, pp. 38, 48). As discussed above, this is incorrect; IBM does not in fact disclose or render obvious a printer with this functionality.

Regarding Ground No. 9, the Examiner incorporated the claims chart of Requester's Exhibit L by reference. However, the claim chart for claim 20 blandly states that the Interleaf Patent is "directed to printing," and that the IBM reference "teaches a printer." First, the Interleaf Patent is not directed to printing; it is directed to editing a document at a workstation, which performs the purported object processing. The printing functionality is ancillary, and there is no disclosure of performing object processing at a printer. Likewise, as discussed above, the IBM reference makes no such disclosure. Requester does not even allege that it would have been obvious to perform the methods of the Interleaf Patent, or even IBM, in a printer.

Indeed, it is instructive to compare the rejection of claim 20 with that of claim 21 in this regard. With respect to claim 21, directed to a printer server, Requester states that "[i]t would have been obvious to implement the system of the Interleaf Patent as a 'print server.' Running printer software on a printer server is 'merely a matter of obvious engineering choice,' and thus prima facie obvious." No such statement has been made (or can be made) about implementing the system of claim 17 on a printer, and in fact, Requester does not even attempt to do so.

Additionally, claim 59 has been rejected on Ground No. 7 over Interleaf in view of the Interleaf Patent. For this claim, the Examiner states that the claim is rejected on the same basis as claim 20, which is not rejected on Ground No. 7 (ACP, p. 63). The portion of the ACP discussing claim 59 does not even mention the word printer. The ACP also incorporates by reference Requester's Exhibit K, which does not include a claim chart for claim 59. The claim chart for claim 20 (which is not rejected on the basis of Interleaf in view of the Interleaf Patent) states that Interleaf discloses a printer, and that the Interleaf Patent is directed to printing documents (Exhibit K, p. 46). However, there is nothing to show that Interleaf or the Interleaf Patent disclosed or rendered obvious the printer recited in claim 20 or 59, i.e., a printer having the particular functionality recited in the claims.

Third Party Requester notes (pp. 2-4) that Patent Owner makes two separate arguments with regard to these claims, addressed below:

The PO makes several arguments (PO Remarks at 10-12) that claims 20, 59-63, and 64-67 are directed to a "printer" with the functionality described in claims 1 and 17. First of all, the Requester notes the above discussion of these claims with reference to the section 112 rejections. Secondly, this is still taught by the prior art. IBM teaches that you can have both "system-attached" printers and "network printers." IBM at 6. Specifically, chapter 2 of IBM is directed to "Printing on the AS/400," and states:

> "AS/400 printing was formerly limited to a single print data format and was printed only on twinaxial-connected line printers. Print support in the AS/400 operating system consisted of a single printing subsystem that supported all print application output, spool management, and printer devices. Most AS/400 output was generated in the SNA

> Character String (SCS) print-data format and was printed on SCS line printers ...." "AS/400 now provides a wide range of printing support including AFP printers of various technologies, speeds, and capabilities. AS/400 has also extended print support to the network, enabling it to serve clients and printers on the network." IBM at 11, emphasis added.

Thus, the AS/400 has an attached printer already, and the IBM reference teaches how the AS/400 can also provide a print data stream to printers on the network. This is further shown in Fig. 178 at page 289 of IBM, reproduced and annotated below. The AS/400 with its attached line printer combined are "a printer" (highlighted in the figure with a dotted box) that outputs a "print data stream" to a printer (one of the two networked printers on the right side of the figure). Thus, IBM reads on the claims.

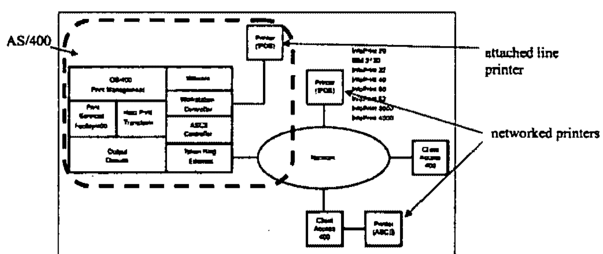


Figure 178. Printing LAN applications on workgroup printers
IBM, p. 268

Likewise, the Interleaf patent is directed towards "a Document Processing System for creating, editing, printing and presenting active electronic documents." 1:8-11. Thus, the "system" of the Interleaf patent is for "printing," and can therefore be considered a "printer."

Finally, the background of the '789 patent shows that intelligent printers (i. e., printers with processing power) were not new and that it was known in the art to put processing functionality on printers. See, e.g., 1:17-21; 2:44-48. The purported novelty of this patent was not to put the processing on a printer as opposed to a computer; it was the processing method itself. See 2:62-

3:4.

The **Requester** has shown that the method was not novel; executing the method on the printer itself

(rather than a computer) does not make the method non-obvious, but instead adheres to the trend towards intelligent printers as described in the background section of the '789 patent. It is noted that the claims do not recite that the system and first printer is contained in a single, common chassis.

Examiner asserts that the IBM reference teaches an AS/400 with the system or functionality as recited in claims 17 and 1 The AS/400 performs print data stream processing, using DDS scripts, and sends the output stream to the printer. The AS/400 may act as a printer (as shown in IBM diagram above) and includes a CPU programmed with functionality that can transform digital print data streams.

It is noted that at times Patent Owner appears to refer to a printer as merely a printer device, and at other times infers that the claimed inventive printer is a device with CPU programmed functionality. Whether a "printer" is housed within a common chassis or is housed external (not within a common chassis) to a linked CPU programmed with functionality is insignificant. See *In re Larson*, 340 F.2d 965, 968, 144 USPQ 347, 349 (CCPA 1965). (A claim to a fluid transporting vehicle was rejected as obvious over a prior art reference which differed from the prior art in claiming a brake drum integral with a clamping means, whereas the brake disc and clamp of the prior art comprise several parts rigidly secured together as a single unit. The court

affirmed the rejection holding, among other reasons, "that the use of a one piece construction instead of the structure disclosed in [the prior art] would be merely a matter of obvious engineering choice.")

The prior art combinations teach a printer device linked to a CPU programmed with the functionality as disclosed in claim language. As noted by Third Party Requester, IBM, Figure 178 shows claimed limitations. The AS/400 (data processing unit, memory, communications interface) has an attached printer already, and the IBM reference teaches how the AS/400 can also provide a print data stream to printers on the network (an output data stream).

See *In re Fridolph*, 309 F.2d 509, 50 CCPA 745. (When determining the obviousness of an invention.) Kahn, 441 F.3d at 986 ("[T]he 'motivation-suggesting-teaching' requirement protects against the entry of hindsight into the obviousness analysis."); *In re Fridolph*, 30 CCPA 939, 942 (1943) ("[I]n considering more than one reference, the question always is: does such art suggest doing the thing the [inventor] did."). According to the "motivation-suggesting-teaching" test, a court must ask "whether a person of ordinary skill in the art, possessed with the understandings and knowledge reflected in the prior art, and motivated by the general problem facing the inventor, would have been led to make the combination recited in the claims." *Kahn*, 441 F.3d at 988 (citing Cross Med. Prods., Inc., v. Medtronic Sofamor Danek, Inc., 424 F.3d 1293, 1321-24 (Fed. Cir. 2005)).

Examiner disagrees with Patent Owner and asserts that the Interleaf Patent is directed many aspects of document processing, including printing ("a Document Processing System for creating, editing, printing and presenting active electronic documents," 1:8-11). The combination of Interleaf Patent and IBM reference do disclose performing object processing at a printer.

Examiner notes that original claim 20 and new claim 59 are rejected as anticipated by IBM. Alternately, original claim 20 and new claim 59 are rejected (ACP 04/09/2012) under combinations comprising IBM, Interleaf and/or Interleaf Patent. (New claim 59 is additionally rejected under the combination of Interleaf and Interleaf Patent.) As **Examiner** has pointed out above, IBM, alone or in combination with Interleaf and /or Interleaf Patent teaches the claim limitations. See ACP 04/09/2012, pp. 19-23.

Examiner agrees with the point made by Third Party Requester: The background of the '789 patent shows that intelligent printers (i. e., printers with processing power) were not new and that it was known in the art to put processing functionality on printers. See, e.g., 1:17-21; 2:44-48. The purported novelty of the '789 patent was not to put the processing on a printer as opposed to a computer; it was the processing method itself. See '789, 2:62-3:4: "It is therefore an object of the present invention to specify a method for the transformation of digital print data streams which is both capable of recognizing more complex page description languages, whose syntax may no longer be described with the aid of simple regular expressions, and also provides the preconditions that the recognized graphic objects can be transformed into a target format, but

also processed further, as flexibly and effectively as possible, that is to say with regard to their description at the highest possible level of abstraction."

2. Patent Owner asserts (PO Remarks 05/09/2012, pp. 12-16), "There is a Clear Nexus Between the Commercial Success and the Printer Claims." Examiner has not made a prima facie case of anticipation or obviousness...the objective evidence shows resoundingly that the inventions covered by the printer claims in particular were not obvious. **Patent Owner** has submitted extensive evidence of commercial success, praise in the industry, and copying, and particularly, the undisputed facts of the Widuch declaration and its exhibits, all of which are incorporated herein by reference. Regarding commercial success, the Widuch Declaration showed that:

> * Konica-Minolta - the printer company - paid IBM millions of dollars for use of the technology in its printers and agreed to pay a running royalty to IBM, of which CCP received 50%. (Widuch Dec. ¶ 8).
> * Samsung paid IBM several million as a lump sum and was supposed to pay IBM a running royalty for each device that included the patented JScribe® technology. (Widuch Dec. ¶ 10). As shown in the White Paper, discussed below, these devices were all multifunctional printers (MFP's).

Regarding praise in the industry, the exhibits showed that:

> * IBM and Samsung issued a joint press release that stated: " 'JScribe', the printing optimized middleware of IBM will be launched in the Samsung digital printing devices such as printers and multifunction products." (Widuch Dec. Ex. B, emphasis added).
> * Samsung's 2008 press release was specifically directed to "the launch of the SCX-5835FN monochrome laser multifunction printer (MFP)." (Widuch Dec. Ex. E, p. 1, emphasis added). The press release further states that "[c]orporate users can maximise their printing investment by customising the device to their specific requirements. For example, users can incorporate additional functionality such as security solutions or job accounting. Samsung's JScribe TM open platform technology means these new features

> can be added quickly and easily in response to the individual needs of the customer." (Widuch Dec. Ex. E, p. 2, emphasis added).

Most telling is Samsung's effusive praise for the patented JScribe® technology and its printer applications in a white paper issued in about 2008-09 (Widuch Dec. Ex. F). The white paper dealt exclusively with the installation and uses of the patented invention in multifunctional printers: [t]his document is intended for those who want to understand JScribe and its uses for Samsung B2B multifunctional devices (MFDs)." (Widuch Dec. Ex. F, p. 1). There are numerous references to the benefits of the Jscribe technology as embedded in printers:

- "JScribe is an embedded application and communication platform for MFP's [multifunctional printers], and it has been specifically designed to provide a custom application suite for these printers." (p. 1, emphasis added);
- "User interface objects allow interaction between the user and the MFP control panel and keypad. JScribe has been designed to allow low level access to the control panel and keypad. By using the built-in JScribe panel functionality, developers can create custom applications for the control panel and keypad on the MFP." (p. 1, emphasis added);
- "Print job filter objects allow JScribe applications to recognize incoming print data and, if required, modify the print sequences before the print data reaches the MFP's RIP [Raster Image Processor]." (p. 2, emphasis added).
- "In the past, modifying the printer code was either impossible or extremely expensive. This left the user with very few choices when a new printer-related business task emerged." (p. 3, emphasis added).

The document continues, explaining in detail the particular benefits of installing Jscribe in printers:

> Most MFPs provide a set of built-in features, which allow users to perform complex tasks such as, scanning directly to Email or scanning directly to an FTP site. These built-in features are hardcoded into the printer. In the past, modifying the printer code was either impossible or extremely expensive. This left the user with very few choices when a new printer-related business task emerged. They could make expensive modifications to their existing MFPs, or they could make a capital investment in new MFPs, which could become obsolete with the discovery of the next new printer related business task.

> Now JScribe allows users to easily customize and enhance JScribe enabled Samsung MFPs to meet their new business needs as they arise. The JScribe solutions are as easy to install as new software programs on a PC. And JScribe currently includes many MFP solutions, with new features being developed continually for Samsung. (p. 3)

Finally, the examples provided in the "Integration" section of the White Paper as innovative applications of Jscribe - the Automated Document Distribution; Stand-Alone Kiosk Copyshop; Automated Workflow Processes; Variable and Secure Form Processing; High- Security Printing & Scanning - were all for the use of the patented Jscribe software in printers. (pp. 5-6). It is also worth noting, as to commercial success, that just these exemplary applications show deployment of approximately 8,400 printers having Jscribe installed therein. Jscribe was not a standard feature of the printer; customers opted (and paid) to install or enable Jscribe on their printers separately from purchase of the printer itself. Finally, regarding copying, Samsung has not contradicted that it copied the Jscribe software in its printers.

Therefore, there is a clear nexus between the particular inventions of the printer claims and the objective evidence of non-obviousness, including commercial success, praise in the industry, and copying. Objective evidence of non-obviousness must be considered. *Transocean Offshore Deepwater Drilling, Inc. v. Maersk Contractors USA, Inc.,* 617 F.3d 1296, 1305 (Fed. Cir. 2010) ("IT]he district court must *always* consider any objective evidence of nonobviousness presented in a case.") (Emphasis in original); see MPEP § 716.01(a) ("Affidavits or declarations... containing evidence of criticality or unexpected results, commercial success, long-felt but unsolved needs, failure of others, skepticism of experts, etc., must be considered by the examiner in determining the issue of obviousness of claims for patentability...") (Emphasis added); § 2154

("Evidence pertaining to secondary considerations must be taken into account whenever present[.]") (Emphasis added).

Evidence of secondary considerations may often be the most probative and cogent evidence in the record. It may often establish that an invention appearing to have been obvious in light of the prior

art was not. It is to be considered as part of all the evidence, not just when the decision maker remains in doubt after reviewing the art.

Stratoflex, Inc. v. Aeroquip Corp., 713 F.2d 1530, 1538 (Fed. Cir. 1983) (emphasis added, internal quotation marks omitted). "Once the applicant has presented rebuttal evidence, Office personnel should reconsider any initial obviousness determination in view of the entire record." MPEP § 2141(V).

Indeed, the objective evidence of non-obviousness may overcome a prima facie case of obviousness. *Gambro Lundia AB v. Baxter Healthcare Corp.,* 110 F.3d 1573, 1580 (Fed. Cir. 1997) (reversing finding of invalidity where district court failed to consider objective evidence of non-obviousness, including commercial success and failure of others); *Ruiz* at 667; *Hughes Tool Co. v. Dresser Industries, Inc.,* 816 F.2d 1549 (Fed. Cir. 1987) (affirming validity based on objective evidence of non-obviousness); *Simmons Fastener Corp. v. Ill. Tool Works', Inc.,* 739

F.2d 1573, 1575, (Fed. Cir. 1984) (reversing holding of invalidity and finding that the objective evidence overcame the lower court's decision on obviousness, holding that "[o]nly after all evidence of nonobviousness has been considered can a conclusion on obviousness be reached."). Samsung's use of the patented technology certainly demonstrates commercial success; Samsung's and IBM's comments show praise in the industry; and Samsung's illegal use following termination of the IBM License and its appropriation of the name "JScribe" also shows copying, all important objective of non-obviousness that easily overcome any *prima facie* claim of obviousness. Moreover, all of these objective indicia relate specifically to printers installed with the patented software, i.e., there is a nexus between the evidence and the printer claims in particular. This evidence must be considered in any obviousness analysis, *Transocean* at 1305 (Fed. Cir. 2010); see MPEP §§ 716.01(a), 2154, and changes the result here.

Third Party Requester (pp. 4-5) rebuts that Patent Owner's recitation of "objective evidence" simply repeats the arguments it already made and which the ACP already rejected. The Examiner deemed that the evidence was "insufficient to rebut the prima facie case of obviousness." ACP at 165, see also, MPEP 716.01 (d)("Although the record may establish evidence of secondary considerations which are indicia of nonobviousness, the record may also establish such a strong case of obviousness that the objective evidence of nonobviousness is not sufficient to outweigh the evidence of obviousness.") The prior art in the present request is extremely strong, and actually anticipates 30 of the 53 claims, including all of the independent claims. In addition to the Examiner's reasons, the evidence also fails because the PO has not established a nexus between the evidence and the claimed invention. MPEP § 716.01 (b). In

particular, the PO has not addressed the fact that the licenses for JScribe® were all software licenses, where actual source code was provided. Actual source code provides a commercial benefit to the licensees by itself, without regard to any patent license. The PO has not asserted that the '789 patent was even mentioned in the licenses. Nor has the PO shown that the licensees entered into the licenses in order to obtain the allegedly patentable aspects of the source code. See EWP Corp. v. Reliance Universal Inc., 755 F.2d 898, 907-08 (Fed. Cir. 1985):

Such [licensing] programs are not infallible guides to patentability. They sometimes succeed because they are mutually beneficial to the licensed group or because of business judgments that it is cheaper to take licenses than to defend infringement suits, or for other reasons unrelated to the unobviousness of the licensed subject matter.

Samsung has copies of the IBM and Samsung licenses, and the '789 patent is not mentioned. CCP did not include the Konica Minolta license in its papers. In its revised Remarks, CCP cites an additional 2004 agreement with IBM. (PO Remarks at 2; Picht Dec. ¶ 7). But that agreement was also a software license that did not specifically mention the '789 patent. Thus, consistent with the Examiner's analysis in the ACP, any alleged "objective evidence" is insufficient to overcome the adopted obviousness rejections.

Examiner notes that Patent Owner arguments are repetitive. See ACP (04/09/2012, pp. 79-90) regarding Examiner's consideration of secondary evidence. Objective evidence of non-obviousness has been considered. **Examiner** hereby reconsiders and maintains the initial strong

case of an obviousness determination in view of the entire record. **Examiner** notes that Patent Owner has confirmed that known prior art teaches printers enhanced with code, and that Patent Owner asserts that modifying the printer code was difficult and / expensive (See excerpt from Exhibit F, p. 3 above.). See '789 1: 17-42, 2: 44-48. A stated objective of the invention ('789, 2: 62-3:4) is transformation of digital print data streams capable of recognizing more complex page description languages, whose syntax may no longer be described with the aid of simple regular expressions, and also provides the preconditions that the recognized graphic objects can be transformed into a target format, but also processed further, as flexibly and effectively as possible, that is to say with regard to their description at the highest possible level of abstraction.

As previously noted in ACP (04/09/2012), the Widuch Dec, Exhibit F is undated. As noted in ACP (pp. 79-90), Patent Owner has not established a nexus between the evidence and the claimed invention. MPEP § 716.01 (b). There are no software licenses included in Patent Owner exhibits. Patent Owner has not shown that success is linked to the claimed invention and not to some other factor, e.g., some other feature of the item sold, advertising / information on advertising within the relevant market, sales strategy / a description of the relevant market for the product. Patent Owner has not shown sales results including total sales for competing products in the market, indicated differences between these products and the Patent Owner's product, shown total sales for products embodying the invention, and pricing of the various products. Merely showing that there was commercial success of an article which embodied the invention is not sufficient. See: *Ex parte Remark*, 15 USPQ2d 1498 (Bd. Pat. App. & Inter. 1990). Gross sales figures do not show commercial success absent evidence as to market share. See: *Cable*

Electric Products, Inc. v. Genmark, Inc., 770 F.2d 1025, 226 USPQ 881 (Fed. Cir. 1985) Inventor's opinion as to the purchaser's reason for buying the product, alone, is generally insufficient to demonstrate a nexus between the sales and the claimed invention. See: *In re Huang,* 100 F.3d 135, 40 USPQ 2d 1685 (Fed. Cir. 1996). **Examiner** maintains that the record has established a strong case of obviousness such that any alleged objective evidence of nonobviousness is insufficient to outweigh the evidence of obviousness.

B. Patent Owner argues (pp. 16-18) the Interleaf, Interleaf Patent, and Lieberman references disclose editing a document on a computer screen, while claim 19 recites a system for the transformation of digital print data streams that allows stored objects to be graphically manipulated and transformed automatically into Java Script objects, in which the manipulations are "transformed automatically into Java Script objects." "The Interleaf, Interleaf Patent, and Lieberman references are irrelevant to print streams"..."none of the cited references disclose every element of claim 19."

Regarding Ground No. 8 (Interleaf in view of Interleaf Patent and Lieberman), and Ground No. 9 (Interleaf Patent in view of IBM) the ACP... incorporates by reference Requester's Exhibit K / incorporates by reference Requester's Exhibit L. Claim element [19c] recites that the data processing unit permits objects "to be read out graphically, to be changed, to be deleted or to be appended, these graphically performed manipulations if necessary transformed automatically into Java Script objects." For this claim element, Requester cites the Interleaf Patent, as disclosing a system that allows a user to edit a document by "modifying existing text and

graphics or by adding entirely new text or images." (Exhibit K, p. 46, citing Interleaf Patent, Col. 1, lines 15-27 / (Exhibit L, p. 36, citing Interleaf Patent, Col. 1, lines 18-27).

The cited portion of the Interleaf Patent discloses:

> Electronic Document Processing Systems generally include a computer workstation programmed to allow a user to create and edit an electronic representation of a document. The workstation includes a display device for displaying an image of the document as it would appear if printed. For example, a document such as a magazine article may contain an arrangement of text and graphics. The system allows the user to edit the document by modifying existing text and graphics or by adding entirely new text or images. As the user modifies the document, the display is continually updated to reflect these changes, thereby allowing the user to interact with the system to achieve the desired document. (Interleaf Patent, Col. 1, lines 15-27).

Nothing in the above portion (or elsewhere) in the Interleaf Patent discloses "these graphically performed manipulations.., being transformed automatically into Java Script objects," as recited in claim 19. At most the Interleaf Patent describes the Lisp environment, but there is no disclosure of this element of claim 19. Accordingly, claim 19 should be confirmed.

Third Party Requester (pp. 5-6) rebuts Patent Owner arguments (PO Remarks at 16-18, Interleaf, Interleaf Patent, and Lieberman references are irrelevant to print streams) and asserts that "It is true that these references discuss editing a document on a computer screen, but they also unequivocally discuss print streams as well." For example, Interleaf teaches: "If a document contains active objects within the document file stream ..., these become activated when the document is opened programmatically or by the user for ... printing." Interleaf, p. 84, emphasis added. The Interleaf patent states: "This invention relates to a Document Processing System for

creating, editing, printing and presenting active electronic documents which are associated with computer programs." Interleaf patent at 1:8-11, emphasis added.

Requester rebuts Patent Owner arguments (PO Remarks at 18.) directed to the Interleaf patent and "transformed automatically into Java Script objects." **Requester** asserts that claim 19 is not limited to Java Script, but instead identifies Java Script as a preference, or possibility. Specifically, the claim states "preferably also Java Script objects" and "if necessary being transformed automatically into Java Script object." Furthermore, the term "JavaScript" is a registered trademark of the Oracle Corporation,[3] and as such, identifies a source of goods, not the goods themselves. As stated in MPEP 2173.05(u): "It is important to recognize that a trademark or trade name is used to identify a source of goods, and not the goods themselves. Thus a trademark or trade name does not identify or describe the goods associated with the trademark or trade name." Thus, Java Script is not a limitation to the claim.

Requester further states, "The Interleaf patent describes automatic transformation into LISP, as described in the PO's citation and admitted by the PO. Specifically, when a user modifies a document, the corresponding objects are automatically updated to reflect the changes. Interleaf patent, 1:15-27. It is noted that the proposed rejection relies on Lieberman to show that Java Scripts were a known alternative to LISP." "One cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references." MPEP 2145.IV.

Examiner agrees with Requester that Interleaf, Interleaf Patent, and Lieberman references discuss print streams. **Examiner** agrees that Requester's cite to Interleaf, p. 84 and Interleaf Patent at 1: 8-11 fairly teach the subject matter of claim 19, element c. **Examiner** asserts that the combination of prior art references disclose every element of claim 19. Patent Owner arguments directed towards Java Script objects are not on point with claim language. **Examiner** asserts that Patent Owner cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references." **Examiner** agrees with Requester that the term "JavaScript" is a registered trademark and that a trademark or trade name is used to identify a source of goods, and not the goods themselves and does not identify or describe the goods associated with the trademark or trade name. Interleaf's active objects within the document file stream fairly read on the "objects" of claim 19, element c.

C. **Patent Owner** argues (pp. 18-22) the rejections of claims 55, 63, 67, and 75:

Claim 55 recites the system for the transformation of digital print data streams according to claim 54 "characterized in that the data processing unit is further programmed to send and receive e-mails in the cases defined in the script."

Claim 63 recites the printer according to claim 59, "wherein said printer is further adapted to send and receive e-mails in the cases defined in the script."

Claim 67 recites the printing system according to claim 64, "wherein said printer is further adapted to send and receive e-mails in the cases defined in the script."

Claim 75 recites the system according to claim 72, wherein "said printer server is further adapted to send and receive e-mails in the cases defined in the script."

In the ACP, the Examiner rejected claim 55 under Grounds No. 10 (Interleaf in view of the Interleaf Patent and the '705 Patent), No. 11 (IBM in view of Interleaf, Interleaf Patent, and the '705 Patent), and No. 13 (Interleaf Patent in view of IBM and the '705 Patent). The Examiner also rejected claims 63 and 67 on Ground No. 13. The ACP does not set forth the Ground of Rejection of claim 75; however, Patent Owner presumes it was intended to be included in Ground No. 13.

The '705 Patent discloses an architecture for document archival:

> Archiving and retrieving and classifying documents into meaningful collections is accomplished in a manner similar to sending email to recipients, retrieving messages from folders, and classifying messages into folder hierarchies. In the simplest scenario, if saveme.com is the archiving server's name, then sending an email to abc@saveme.com will cause the contents of the email message to be archived in the abc mailbox. The archived documents may be automatically stored in jukeboxes of non-tamperable media such as Write Once Read Multiple (WORM) Compact Disks (CD), which provide high storage capacity, low cost compared to magnetic disks, random data access, and long-term stability. The present invention leverages existing messaging infrastructures, and the resulting environment is not intrusive, easier to administer, and easier to deploy than conventional dedicated document archival systems. (Abstract)

The '705 Patent discloses that in order to archive a document, a user may send the document as an attachment to an email addressed to a public folder, e.g., send to folder "abc" by emailing the document as an attachment to abc@saveme.com. (Col. 3, lines 18-22). The handling of such an email to the folder is performed by means of a server-side script associated with an event. The

script collects metadata of the received message, possibly extracts information from the body of the email, and stores the document. (Col. 3, lines 39-62).

In Ground 10, the Examiner stated that it would have been obvious to combine the teachings of the '705 Patent with Interleaf and Interleaf Patent. As discussed above, Interleaf and Interleaf Patent are in the field of document editing. Even according to the Examiner's definition of the relevant field, Interleaf and Interleaf Patent are in the field of "object-oriented document publishing systems with scripting functionality." (ACP, p. 33). However, the '705 Patent has nothing to do with printing or publishing. Only with clear hindsight would it have been obvious to modify the Interleaf references relating to document editing with the '705 patent's document archiving. Should the field, then, be defined as simply "scripting functionality"? In any event, if combined, the '705 Patent and Interleaf reference would merely allow an "active" document created and edited using Interleaf to be archived by emailing such a document as an attachment to a particular email address. However, this combination of the '705 Patent and Interleaf references is not the invention recited in claim 55. First, even Requester does not suggest that an email, which is what is read by the script in the '705 Patent, is a print data stream. The '705 Patent does not disclose analyzing the input print data steam by means of a parser for graphically representable objects or split up the input print data stream into these graphically representable objects. The '705 Patent does not transform the graphically representable objects into a format for the control of a printer, nor does it combine the objects into an output print data stream, and output said combined output print data stream to said printer.

Second, and more importantly, the script disclosed in the '705 Patent is not sent by a script. On the contrary, in the '705 Patent, a script is triggered by an email. In contrast, claim 55 of the '789 Patent recites the causal opposite - an email is triggered by a script. Nothing in the '705 Patent or its combination with the other references would have suggested or motivated this functionality.

Finally, the '705 Patent does not disclose sending emails at all, much less, by a script. Rather, the '705 Patent only discloses a user sending the email. Nothing in the combination of the '705 Patent and the Interleaf references suggests a data processing unit programmed to send and receive e-mails in the cases defined in a script.

Regarding Ground No. 10, the Examiner's motivation for detaching the email scripting functionality of the '705 Patent from the archiving context and implanting it into the document editing functionality of the Interleaf references is particularly unpersuasive. First, as discussed above, the '705 Patent does not disclose email functionality triggered by a script at all, but rather, a script triggered by receipt of an email.

The ACP states that combining Interleaf with the message scripting of the '705 Patent "combin[es] well known printing techniques to yield predictable results." (ACP, pp. 74, 78, emphasis added). However, the '705 Patent does not disclose printing techniques at all. (Nor do the Interleaf references, for that matter.) As for the predictability - computers are inherently

predictable; they perform as programmed. Therefore, once the Examiner has suggested in hindsight to implant a script that sends and receives emails into the Interleaf system, the system will naturally perform the predicted functionality. This is classic hindsight.

The Examiner also points to the statement of the Interleaf reference that it "takes advantage of all the other publishing facilities." (ACP, p. 74). This tenuous connection provides the hook by which the Examiner loops in the '705 Patent; however, this general invitation cannot provide a basis to combine with any and every functionality. Indeed, the '705 Patent's archiving system is not an external publishing facility. Failing these, the Examiner resorts to "common sense", which would supposedly motivate one of ordinary skill to combine the references, on the sole basis that they purportedly have "similar scripted functionality."

Regarding Ground No. 11, the Examiner combines four references (IBM, Interleaf, Interleaf Patent, and the '705 Patent) to reject claim 55. The motivation provided for this combination likewise relies on a flawed definition of the relevant field ("object-oriented document publishing systems"), and clear hindsight. The Examiner cites the IBM reference's marketing statement that AFP "enables you to take full advantage of printing technology," (ACP, p. 76), even though the '705 Patent does not relate to printing technology. The Examiner also referred to a "person investigating the '705 Patent" as being motivated by common sense to find other descriptions of known scripting functionality, which would lead to the combination of IBM, Interleaf, Interleaf Patent, and the '705 Patent. However, a "person investigating the '705 Patent" already assumes

the answer. The Examiner has not asked (much less answered) the question of why a person of ordinary skill in the relevant art would be investigating the '705 Patent to begin with.

As discussed by Dr. Birnbaum, a person of ordinary skill working in the relevant field in May 2000 (the earliest priority date) or as late as May 2001 (the PCT filing date), would have been a person with a computer science degree (typically, a bachelor's degree), and approximately 2-4 years of experience. (Birnbaum Dec. ¶ 58). This experience may typically have been acquired working at a company making printers or printer software, such as Xerox, Hewlett-Packard, etc. (Birnbaum Dec. ¶ 59). The person of ordinary skill in the art is "an objective legal construct presumed to think along conventional lines without undertaking to innovate, whether by systematic research or by extraordinary insights," *Life Technologies, Inc. v. Clontech Laboratories, Inc.*, 224 F.3d 1320, 1325 (Fed. Cir. 2000); *Std. Oil Co. v. Am. Cyanamid Co.*, 774 F.2d 448, 454 (Fed. Cir. 1985) ("A person of ordinary skill in the art is also presumed to be one who thinks along the line of conventional wisdom in the art and is not one who undertakes to innovate, whether by patient, and often expensive, systematic research or by extraordinary insights, it makes no difference which." (Emphasis added)). This person does not look for exceptionally creative solutions, and does not consider how to completely redesign existing software to arrive at novel, and admittedly extremely useful products that were previously unknown. In short, the person of ordinary skill in the art does not "think outside the box."

Under the correct legal standard for avoiding hindsight, none of the motivations provided by the Examiner for combining the '705 Patent with IBM and the Interleaf references (e.g., printing technology, external devices, common sense) can support a *prima facie* obviousness rejection of claim 55.

This rejection of claims 63 and 67 (and presumably, 75) on Ground No. 13 (Interleaf Patent in view of IBM and the '705 Patent) is likewise flawed. In particular, claims 63 and 67 recite a "printer... adapted to send and receive e-mails in the cases defined in the script." None of the Interleaf Patent, IBM, or the '705 Patent disclose or suggest providing a printer having scripting functionality. It is therefore not disclosed, nor would it have been obvious, to provide a printer with the emailing functionality recited in claims 63 and 67.

In addition, the objective evidence discussed above is likewise applicable to the email functionality. For example, the White Paper touts the particular benefits of Jscribe as follows: "MFP features you can control with JScribe include the following: ... The MFP can output to FTP or Email." (Widuch Declaration, Exhibit F, p. 3, emphasis added). If this feature was so obvious, it would hardly be worth mentioning in the White Paper.

Third Party Requester (pp. 6-7) rebuts Patent Owner arguments regarding claims 55, 63, 67, and 75 (PO Remarks at 18-22) and asserts that Patent Owner improperly argues the references

individually. The '705 patent shows that it was known in the art to receive an email message via a script:

> [T]he user sends the document to be archived as an attachment to an email addressed to the public folder that should contain the document, e.g. abc@saveme.com, where abc is the name of a public folder on the archiving server saveme.com 120 .... the deposit of a message in a folder triggers the Folder: :OnMessageCreated event. As a result of the triggered event, a server-side script associated with this event is invoked and is passed the folder identification and the message identifier of the newly posted message. The script at step 122 first collects the properties of the message (date, sender, etc.) that can be automatically derived and assigns a unique id to the document. Then, it parses the body of the message and, if the body of the message is a form, it extracts all information contained in the form. '705 patent at 3:17-22, 40-50.

The PO next argues that "in the '705 Patent, a script is triggered by an email. In contrast, claim 55 of the '789 Patent recites the causal opposite - an email is triggered by a script." (PO Remarks at 20, emphasis in original.) Again, the '705 patent is being used to show that it was known in the art to receive an email message via a script. As for triggering an email by a script, the rejection relies on the Interleaf Patent. The Interleaf Patent teaches a "WYSIWYG document that can send itself over an electronic mail system." Interleaf Patent at 7:50-51; *see also,* 8:19-22. The PO does not contest the fact that the Interleaf Patent teaches sending (or triggering) an email by a script.

The PO's third and final argument regarding this issue is that "the '705 Patent does not disclose sending emails at all, much less, by a script. Rather, the '705 Patent only discloses a user sending the email." PO Remarks at 20. Again, the rejection relies on the Interleaf Patent to show that it was known in the art to send emails by a script. Thus, receiving email messages via a script is shown by the '705 patent and sending email messages via a script is shown by the

Interleaf patent. As for Ground No. 10, the PO argues against the combination of the '705 patent and the Interleaf references. The Interleaf references and the '705 patent are all directed to printing documents:

- Interleaf: "If a document contains active objects within the document file stream (as opposed to the methods file), these become activated when the document is opened programmatically or by the user for viewing, editing, or printing." P. 84, emphasis added.
- Interleaf patent: "This invention relates to a Document Processing System for creating, editing, printing and presenting active electronic documents which are associated with computer programs." 1:8-11, emphasis added.
- '705 patent: "Messaging is also used as the middleware for accessing and controlling shared resources, such as a printer server that receives documents as electronic mail (email) and prints a document at the appropriate printer based on the available printers, the document type, and the sender's identity." 1:66-2:4, emphasis added.

These documents all describe well-known aspects of document printing, and "[a]pplying a known technique to a known device (method, or product) ready for improvement to yield predictable results" is a proper rationale for supporting a finding of obviousness under section 103. MPEP 2143, citing *KSR International Co. v. Teleflex Inc.*, 550 U.S. 398,417-20 (2007).

Examiner agrees with Requester. The claim limitation, "receiving email messages via a script" is taught by the '705 patent and the claim limitation, "sending email messages via a script", is taught by the Interleaf patent. Regarding the combination of the '705 patent and the Interleaf references, the Interleaf references and the '705 patent are all related to printing documents. See Requester's citations in discussions above.

Regarding the rejection of new claim 75 (See PO comment, PO Remarks 05/09/2012, p. 18), attention is directed to the ACP (04/09/2012, p. 22, last paragraph) "New claim 75 is rejected

under Ground #11 (IBM in view of Interleaf, the Interleaf Patent, and the '705 patent). Pertinent discussions in Requester Comments (09/06/2011), pp. 24-32 are hereby incorporated by reference. Patent Owner should understand that new claim 75 is rejected for the same reasons as claim 55:

> New **claim 55.** Claim 55 is very similar to issued claims 5 and 8, except that it is limited specifically to sending and receiving e-mails. Sending and receiving emails in response to scripts was well known in the art. For example, the Interleaf Patent teaches a "WYSIWYG document that can send itself over an electronic mail system." (Interleaf Patent at 7:50-51; *see also,* 8:19-22.)]. Similarly, U.S. Patent No. 6,678,705 teaches receiving email messages via a script:
>
> [T]he user sends the document to be archived as an attachment to an email addressed to the public folder that should contain the document, e.g. abc@saveme.com, where abc is the name of a public folder on the archiving server saveme.com 120 .... the deposit of a message in a folder triggers the Folder:: OnMessageCreated event. As a result of the triggered event, a server-side script associated with this event is invoked and is passed the folder identification and the message identifier of the newly posted message. The script at step 122 first collects the properties of the message (date, sender, etc.) that can be automatically derived and assigns a unique id to the document. Then, it parses the body of the message and, if the body of the message is a form, it extracts all information contained in the form.
>
> (Ex. S at 3:17-22, 40-50.) This shows that sending and receiving email messages via a script were both well-known functions in the prior art well before the priority date of the '789 patent. Therefore, for the same reasons discussed in the Request and the OA with respect to claims 5 and 8, and the discussion above, new claim 55 should be rejected according to Grounds 10, 11, and 13. (ACP, 09/06/2011, p. 24)

Patent Owner has admitted in the Background Section of the '789 Specification that it was known for printing processes to be controlled using processor powers, storage capacities and additional options. Applying known techniques to a known device (method, or product) ready for improvement to yield predictable results" is a proper rationale for supporting a finding of obviousness under section 103. MPEP 2143, citing KSR International Co. v. Teleflex Inc., 550

U.S. 398,417-20 (2007). Examiner has determined that one skilled in the art, with the noted references before him, could have made the combination of elements claimed without the exercise of invention. *In re Shaffer*, 229 F.2d 476, 43 C.C.P.A., Patents, 758. Such a combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results. See, *e.g., United States* v. *Adams*, 383 U. S. 39 . When a work is available in one field, design incentives and other market forces can prompt variations of it, either in the same field or in another. If a person of ordinary skill in the art can implement a predictable variation, and would see the benefit of doing so, §103 likely bars its patentability. Moreover, if a technique has been used to improve one device, and a person of ordinary skill in the art would recognize that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond that person's skill. **Examiner** disagrees that hindsight was required to arrive at the claimed invention.

Further, **Examiner** notes that regarding the expert opinion, Exhibit F, attached to the Widuch Declaration and referred to as the "White Paper") is undated.

D. Patent Owner argues (pp. 22-24) the rejections of **claims 56, 62, 66, and 74.**
Claim 56 recites that the data processing unit is programmed "to transform the graphically representable objects into a format for the control of a printer and combine the objects into an output print data stream without a printer driver." Claims 62, 66, and 74 recite similar claim elements. The Examiner rejected claims 56 and 62 on Grounds No. 1, 2, 3, and 9; claim 66 on Grounds No. 3 and 9; and claim 74 on Ground No. 3 alone.

Regarding Grounds No. 1, 2, 3, and 9, the only reference the Examiner cited for the claim element was the IBM reference, and specifically, the portion that teaches "GPT converts the AFP data stream into GP PCL or Lexmark PPDS print data streams. This transform works in one of two ways: raster or mapping." (ACP, p. 30). The Examiner states that "[b]uilding the 'image of the page' in raster mode shows the printing of a high-level print data stream.., into a low-level bitmap image.., for printing on a less-capable printer..." (ACP, p. 30). This rejection is mistaken, since this functionality of the AS/400 that the Examiner points to is precisely the essence of a printer driver. The Examiner has therefore pointed to the very portion of the IBM reference that discloses the precise opposite of claim 56: by building the 'image of the page' in raster mode for printing on a less-capable printer, the AS/400 shows that it includes a printer driver.

Accordingly, where claim 56 (as revised) recites that the data processing unit is programmed "to transform the graphically representable objects into a format for the control of a printer and combine the objects into an output print data stream without a printer driver," the AS/400 is programmed to transform the graphically representable objects into a format for the control of a printer and combine the objects into an output print data stream using a printer driver.

There is also objective evidence of non-obviousness of this feature. The White Paper (p. 2, emphasis added) touts this feature as allowing heightened connectivity:

> Connectivity objects allow JScribe applications to control COM or USB port connected peripheral devices. A simple control script can serve as the device driver and can control a broad range of devices including: - fingerprint readers - Card readers -RFID printers-RFID readers-label printers -Slave printers.

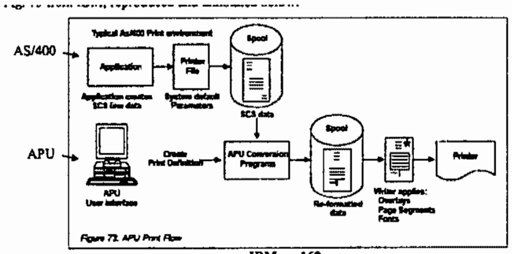
That is, the patented Jscribe script allows control by a printer of other printers without the need for a printer driver. This unexpected benefit is one of the objective indicia of non-obviousness demonstrated in the Widuch Declaration and its exhibits.

Third Party Requester (pp. 7-8) summarizes Patent Owner argument, that the rejections of claims 55, 62, 66, and 74 should be withdrawn because IBM does not teach sending a print data stream without a "printer driver." It is not exactly clear what the PO means by the term "printer driver." Claim 56 recites:

> The system according to claim 54, characterized in that the data processing unit is further programmed to transform the graphically representable objects into a format for the control of a printer and combine the objects into an output print data stream without a printer driver.

The '789 patent provides no indication of what functionality constitutes a printer driver and what does not. Instead, the '789 patent, like both IBM and Interleaf, describes transforming and combining graphically representable objects into a format for the control of a printer. This is the very subject matter of the analysis provided in the base claim 54. In other words, the PO has not shown that the '789 patent does anything different from IBM and Interleaf. Indeed, the PO appears to argue that the processing method in the '789 patent replaces the functionality of a conventional printer driver: "the ability to print without a printer driver is not an independent step, but a result of using the method and system of the invention of the '789 patent." PO Remarks at 4. But as the Examiner concluded with his rejections, IBM and Interleaf teach the same processing method and thus also replace the functionality of a conventional printer driver.

The PO's effort to distinguish the claims from IBM and Interleaf on the presence or absence of a "printer driver" is an exercise in semantics. They all involve the same method for processing print data; whether that is characterized as being a special type of printer driver, or including a printer driver, or replacing a printer driver is irrelevant. Finally, to the extent the Examiner finds that the phrase "without a printer driver" means that the formatting and combining are performed outside of the computer, IBM teaches this same functionality. IBM teaches an Advanced Print Utility (APU) which formats and combines graphical objects. IBM at 267 and 162. The APU operates outside of the AS/400 computer. See Fig. 73 from IBM, reproduced and annotated below.



IBM, p. 162.

Examiner agrees with Requester. **Examiner** agrees that IBM's AS/400 has functionality (scripts) within a processor /data processing unit. **Examiner** agrees that the '789 patent provides no indication of what functionality constitutes a printer driver and what does not. The '789 patent, like both IBM and Interleaf, describe the essence of a printer driver, whereby a data processing unit is further programmed to transform the graphically representable objects into a format for the control of a printer and combine the objects into an output print data stream, i.e. the essence of a printer driver is relocated to within a data processing unit. Patent Owner has not shown that the '789 patent does anything different from IBM and Interleaf.

As discussed above, "The White Paper" (Exhibit F) is undated.

Newly Revised Claim Language

Regarding the revised language of new claims 56, 57, 59, 62, 64, 66, 68, 72, and 74, Patent owner relies on Appendix A (03/20/2011) for support:

56. (New) The system according to claim 54, characterized in that the data processing unit is further programmed to transform the graphically representable objects into a format for the control of a printer and combine the objects into an output print data stream **without a printer driver.**

62. (New) The printer according to claim 59, wherein said printer is adapted to transform the graphically representable objects into a format for the control of an output device and combine the objects into an output print data stream **without a printer driver.**

66. (New) The printing system according to claim 64, wherein said printer is adapted to transform the graphically representable objects into a format for the control of an output device and combine the objects into an output print data stream **without a printer driver**.

74. (New) The printing system according to claim 72, wherein said printer server is adapted to transform the graphically representable objects into a format for the control of a printer and combine the objects into an output print data stream **without a printer driver** associated with the
printer.

Regarding newly revised claim language (claims 56, 62, 66, and 74) and the limitation "without a printer driver," see IBM at p. 270: "HPT converts the AFP data stream into HP PCL or

Lexmark PPDS print data streams. This transform works in one of two ways: raster or mapping. Raster mode builds an image of the page in AS/400 memory to send to the printer ...." Building the "image of the page" in raster mode shows the printing of a high-level print data stream (PCL or PPDS) into a low-level bitmap image (the "image of the page" in raster mode) for printing on a less-capable printer that does not have a PCL or PPDS printer driver specifically installed. The rejections of the ACP are maintained and apply to the newly revised language of claims 56, 62, 66, and 74.

57. (New) The system according to claim 54, further comprising:
an operating station with display means and input means, wherein said operating system makes it possible for the graphically representable objects to be read out via **an application interface**, to be changed, to be deleted, or to be appended before they are output in the output print data stream.

The revised language of claim 57 corrects an antecedent basis problem. The revised limitation, "an application interface" is similarly found in claim 18. Interleaf Patent (1:18-27) teaches a system that allows the user to edit the document (read out via application interface and change...) The rejections of the ACP are maintained and apply to the newly revised language of claim 57.

59. (New) A printer comprising **at least one data processing unit having at least one memory and a communications interface,** said printer adapted for the transformation of digital print data streams comprising:
an input to read an input print data stream;
a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and
a memory to store said graphically representable objects in an object-oriented format,
wherein said printer is adapted to transform the graphically representable objects into a format for the control of an output device, to combine the objects into an output print data stream, and

to output said combined output print data stream, and
wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in cases defined in the script.

Claim language is revised to correct an antecedent basis problem. IBM teaches (p. 1) an AS/400 (Application System/400 with operating system OS/400, p. 400/data processing unit with memory) using an AFP printing and presentation system to transform digital print data streams, to communicate with printers through the Intelligent Print Data Stream (IPDS). The AS/400 (p. 28, interfaces stored in OS) and the printer devices (p. 38) both are data processing units with memory [for the transformation of digital print data streams, data processing unit having memory and communications interface.

Interleaf Patent teaches a data processing unit / programmed computer workstation (FIG. 1A & 1: 15-18) including (2:19-25; a keyboard 16, mouse 18, processor 20, display 14, and memory 12. Interleaf Patent teaches (1:18-27; 5:38-41), "The workstation includes a display device for displaying an image of the document as it would appear if printed (communications interface). For example, a document, such as a magazine article may contain an arrangement of text and graphics. The system allows the user to edit the document (communications interface)."

The rejections of the ACP are maintained and apply to the newly revised language of claim 59.

64. (New) A printing system comprising

a printer comprising **at least one data processing unit having at least one memory and a communications interface,** said printer adapted for the transformation of digital print data

streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer is adapted to transform the graphically representable objects for the control of an output device, to combine the objects into an output print data stream, and to output said combined output print data stream, and

wherein said printer is further adapted to assign at least one script to at least one graphically representable object, wherein said script is to be executed in the cases defined in the script; and

an operating station with display means and input means, wherein said operating station is adapted to allow reading out, changing, deleting, or appending of said at least one script via **an application interface.**

Revised claim language is similar to claim 18 and is taught by the prior art. Interleaf Patent teaches (1:15-27) the system computer workstation (operating station) programmed to allow a user to create and edit an electronic representation of a document. Interleaf Patent teaches (2:19-25) manipulating a keyboard 16, a mouse 18 (input means), using a processor 20 and display 14 (display means) with image data representative of the selected document. In response, display 14 generates an image of the document for viewing by the user." Interleaf Patent (1:18-27) teaches, "The workstation includes a display device for displaying an image of the document as it would appear if printed. For example, a document such as a magazine article may contain an arrangement of text and graphics. The system allows the user to edit the document (read out via application interface and change) by modifying existing text and graphics or by adding entirely new text or images. As the user modifies the document; the display is continually updated to

reflect these changes, thereby allowing the user to interact with the system to achieve the desired document."

The rejections of the ACP are maintained and apply to the newly revised language of claim 64.

68. (New) A printer server comprising at least one **data processing unit having at least one memory and a communications interface,** said printer server for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and
a memory to store said graphically representable objects in an object-oriented format,
wherein said printer server is adapted to transform the graphically representable objects for the control of an output device, to combine the objects into an output print data stream, and to output said combined output print data stream to the output device, and

wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in the cases defined in the script.

IBM teaches (pp. 1, 403, 405) OS/400 (the operating system for the AS/400 processor / data processing unit). IBM teaches obtaining information over the network at pp. 1 and 405: "...to communicate via twinax or to be network-attached via TCP/IP...", "Systems Network Architecture" (SNA) and "TCP/IP." (communications interface)

Interleaf Patent teaches a data processing unit / programmed computer workstation (FIG. 1A & 1: 15-18) including (2:19-25; a keyboard 16, mouse 18, processor 20, display 14, and memory 12 / communications interface).

The rejections of the ACP are maintained and apply to the newly revised language of claim 68.

72. (New) A printing system comprising

a printer server **comprising at least one data processing unit having at least one memory and a communications interface,** said printer server adapted for the transformation of digital print data streams comprising:

an input to read an input print data stream;

a parser to analyze syntax of the input print data steam for graphically representable objects and to split up the input print data stream into these graphically representable objects; and

a memory to store said graphically representable objects in an object-oriented format,

wherein said printer server is adapted to transform the graphically representable objects into a format for the control of a printer, to combine the objects into an output print data stream, and to output said combined output print data stream to the printer, and

wherein said printer server is further adapted to assign at least one script to at least one graphically representable object, wherein said script is executed in the cases defined in the script; and

an operating station with display means and input means, wherein said operating station is adapted to allow reading out, changing, deleting, or appending of said at least one script via an application interface.

The revised language of claim 72 is taught by prior art IBM and Interleaf Patent as noted above in reference to newly revised claim 68. The rejections of the ACP are maintained and apply to the newly revised language of claim 72.

Rejections Summarized

The 35 U.S.C. 112, second paragraph rejections of claims 56, 62, 66, and 74 (as being incomplete for omitting essential steps, such omission amounting to a gap between the steps) are withdrawn.

The 35 U.S.C. 112 second paragraph rejections of claims 57 and 58, based on lack of antecedent basis, are withdrawn.

The 35 U.S.C. 112 second paragraph rejections of claims 59 and 64 (as unclear as to how a printer can output a print data stream), and claims 60-63 and 65-67 due to dependence on claims 59 and 64 are maintained.

The 35 U.S.C. 314(a) rejections of claims 59, 60-63, 68, and 69-71 are withdraw in view of amended claim language (to claims 59 and 68).

The 35 U.S.C. 314(a) rejections of claims 64-67 and 72-75 are maintained. The proposed revised new claim language enlarges the scope of the patented claims because there is no original claim language that recites the scope "A printing system."

Claims 1-64, 66-72, and 74-82 have been reexamined. The rejection of claims 1-64, 66-72, and 74-82 is maintained (incorporated by reference from ACP 04/09/2012). To address the revised claim language, additional citations are noted supra. No claims are found patentable or confirmed.

Service of Papers

Any paper filed with the USPTO, i.e., any submission made, by either the Patent Owner or the Third Party Requester must be served on every other party in the reexamination proceeding, including any other third party requester that is part of the proceeding due to merger of the reexamination proceedings. As proof of service, the party submitting the paper to the Office must attach a Certificate of Service to the paper, which sets forth the name and address of the party served and the method of service. Papers filed without the required Certificate of Service may be denied consideration. 37 CFR 1.903; MPEP 2666.06.

Extensions of Time

Extensions of time under 37 CFR 1.136(a) will not be permitted in these proceedings because the provisions of 37 CFR 1.136 apply only to "an applicant" and not to parties in a reexamination proceeding. Additionally, 35 U.S.C. 314(c) requires that inter partes reexamination proceedings "will be conducted with special dispatch" (37 CFR 1.937). Patent Owner extensions of time in *inter partes* reexamination proceedings are provided for in 37 CFR 1.956. Extensions of time are

not available for third party requester comments, because a comment period of 30 days from service of patent owner's response is set by statute. 35 U.S.C. 314(b)(3).

Notification of Other Proceedings

The patent owner is reminded of the continuing responsibility under 37 CFR 1.985(a) to apprise the Office of any litigation activity, or other concurrent proceeding, involving the patent under reexamination throughout the course of this reexamination proceeding. The third party requester is also reminded of the ability to similarly apprise the Office of any such activity or proceeding throughout the course of this reexamination proceeding. See MPEP §2686 and 2686.04.

Conclusion

This is a RIGHT OF APPEAL NOTICE (RAN); see MPEP § 2673.02 and § 2674. The decision in this Office action as to the patentability or unpatentability of any original patent claim, any proposed amended claim and any new claim in this proceeding is a FINAL DECISION.

No amendment can be made in response to the Right of Appeal Notice in an *inter partes* reexamination. 37 CFR 1.953(c). Further, no affidavit or other evidence can be submitted in an *inter partes* reexamination proceeding after the right of appeal notice, except as provided in 37 CFR 1.981 or as permitted by 37 CFR 41.77(b)(1). 37 CFR 1.116(f).

Each party has a thirty-day or one-month time period, whichever is longer, to file a notice of appeal. This time period may not be extended. 37 CFR 41.61(e). The patent owner may appeal to the Board of Patent Appeals and Interferences with respect to any decision adverse to

the patentability of any original or proposed amended or new claim of the patent by filing a notice of appeal and paying the fee set forth in 37 CFR 41.20(b)(1). The third party requester may appeal to the Board of Patent Appeals and Interferences with respect to any decision favorable to the patentability of any original or proposed amended or new claim of the patent by filing a notice of appeal and paying the fee set forth in 37 CFR 41.20(b)(1).

In addition, a patent owner who has not filed a notice of appeal may file a notice of cross appeal within fourteen days of service of a third party requester's timely filed notice of appeal and pay the fee set forth in 37 CFR 41.20(b)(1). A third party requester who has not filed a notice of appeal may file a notice of cross appeal within fourteen days of service of a patent owner's timely filed notice of appeal and pay the fee set forth in 37 CFR 41.20(b)(1).

Any appeal in this proceeding must identify the claim(s) appealed, and must be signed by the patent owner (for a patent owner appeal) or the third party requester (for a third party requester appeal), or their duly authorized attorney or agent.

Any party that does not file a timely notice of appeal or a timely notice of cross appeal will lose the right to appeal from any decision adverse to that party, but will not lose the right to file a respondent brief and fee where it is appropriate for that party to do so. If no party files a timely appeal, the reexamination prosecution will be terminated, and the Director will proceed to issue and publish a certificate under 37 CFR 1.997 in accordance with this Office action.

All correspondence relating to this *inter partes* reexamination proceeding should be directed:

By U.S. Postal Service Mail to:
Mail Stop Inter Partes Reexam
ATTN: Central Reexamination Unit Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

By FAX to:
(571) 273-9900
Central Reexamination Unit

By hand to:
Customer Service Window
Randolph Building
401 Dulany St.
Alexandria, VA 22314

Registered users of EFS-Web may alternatively submit such correspondence via the electronic filing system at https://efs.uspto.gov/efile/myportal/efs-registered

Any inquiry concerning this communication or earlier communications from the Reexamination Legal Advisor or Examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

/Mary Steelman/

Reexamination Specialist

Art Unit 3992, Central Reexamination Unit

571-272-3704

/Peng Ke/

Reexamination Specialist

Art Unit 3992, Central Reexamination Unit

Conferees:

ALEXANDER J. KOSOWSKI
Supervisory Patent Reexamination Specialist
CRU -- Art Unit 3992

Attorney Docket No.: P-74637-US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patentee(s):	KRAUTTER, Thomas	Examiner:	STEELMAN, Mary J.
Control No.:	95/001,398	Reexamination Filed:	July 16, 2010
Patent No.	6,684,789	Art Unit:	3992
Title:	METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER		

NOTICE OF APPEAL BY PATENT OWNER

Mail Stop Inter Partes Reexam
Central Reexamination Unit
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

The Patent Owner hereby appeals to the Board of Patent Appeals and Interferences on all issues raised by the Right of Appeal Notice dated July 30, 2012 (and to the extent necessary, the Action Closing Prosecution dated April 9, 2012), including the rejection of claims 1-64, 66-72, and 74-82.

The Director is hereby authorized to charge the $620 fee for filing the Notice of Appeal set forth in 37 C.F.R. § 41.20(b)(1), and any additional fees that may be required or associated with this paper, to Deposit Account No. 50-3355.

Respectfully submitted,

/Guy Yonay/
Guy Yonay
Attorney/Agent for Applicant(s)
Registration No. 52,388

Dated: August 27, 2012

Pearl Cohen Zedek Latzer, LLP
1500 Broadway, 12th Floor
New York, New York 10036
Tel: (646) 878-0800
Fax: (646) 878-0801

Attorney Docket No.: P-74637-US

CERTIFICATE OF SERVICE

I hereby certify that on this 27th day of August, 2012, a true and correct copy of the following document:

NOTICE OF APPEAL BY PATENT OWNER

was caused to be served on the following attorney of record:

David L. McCombs
Haynes and Boone, LLP
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Tel: (214) 651-5533

Email: David.McCombs@haynesboone.com

By Email in lieu of First Class Mail (by consent)

/Guy Yonay/
Guy Yonay
Attorney for Patentee
Registration No. 52,388

Dated: August 27, 2012

Pearl Cohen Zedek Latzer, LLP
1500 Broadway, 12th Floor
New York, NY 10036
(646) 878-0800 (phone)
(646) 878-0801 (facsimile)

Attorney Docket No.: P-74637-US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patentee(s):	KRAUTTER, Thomas	Examiner:	STEELMAN, Mary J.
Control No.:	95/001,398	Reexamination Filed:	July 16, 2010
Patent No.	6,684,789	Art Unit:	3992
Title:	METHOD AND SYSTEM FOR THE TRANSFORMATION OF DIGITAL PRINT DATA STREAMS AND CORRESPONDING PRINTER AND PRINTER SERVER		

Mail Stop *Inter Partes* Reexamination
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

PATENT OWNWER'S BRIEF ON APPEAL
UNDER 35 U.S.C. § 134(c) AND 37 C.F.R. § 41.67

Patent Owner and Appellant, CCP Systems AG ("CCP") respectfully submits this Appeal Brief in connection with the above-identified *Inter Partes* Reexamination of U.S. Patent No. 6,684,789 (the " '789 Patent"), as permitted under 35 U.S.C. § 134(c) and 37 C.F.R. § 41.67, and pursuant to the July 30, 2012 Right of Appeal Notice ("RAN").

76. **(New)** The method of claim 1, wherein said input data stream is formatted in a page description language.

77. **(New)** The method of claim 76, wherein said parser is a syntax analyzer, and wherein said analyzing by means of said parser comprises performing syntactic analysis on said input data stream.

78. **(New)** The method of claim 1, wherein storing the graphically representable objects in said object-oriented format comprising storing said graphically representable objects based on membership in hierarchically organized classes.

79. **(New)** The method of claim 1, further comprising dynamically linking a plurality of objects.

80. **(New)** The method of claim 1, wherein the objects are managed by display list management module.

81. **(New)** The method of claim 80, wherein the display list management supports one page and multi-page documents at a plurality of levels.

82. **(New)** The method of claim 81, wherein the display list management can be expanded dynamically by new objects.

IBM teaches DDS keywords BLKFOLD and ZFOLD that control folding systems. IBM at 329, 334.

5. DDS scripts can be applied to Postscript input streams, as recited in claims 76 and 77. CCP notes that IBM discloses converting Postscript data streams into AFP, but then argues that even if the converted AFP stream was parsed, "nothing in IBM allows for a DDS (i.e., a "script") to be assigned to such objects." PO Brief at 20. To the contrary, IBM teaches that once in the AFP format, DDS scripts can be readily applied: "DDS can be used in both the OS/400 printer subsystem and the AFP printer subsystem, but many more formatting capabilities are available with AFP." IBM at 26. Using the example of Figure 116 discussed above, after the input data stream is converted from Postscript to AFP, an OVERLAY script for a letterhead can be applied.

6. IBM teaches a printer that outputs a print data stream to another printer. CCP notes that the AS/400 has an attached line printer, but argues that "the AS/400 and printer are separate devices," and together do not constitute a printer. PO Brief at 22. IBM states:

> AS/400 now provides a wide range of printing support including AFP printers of various technologies, speeds, and capabilities. AS/400 has also extended print support to the network, enabling it to serve clients and printers on the network.

IBM at 11. So, (1) the AS/400 provides printing support including AFP printers, and (2) the AS/400 supports printers on the network. Since the AS/400 includes line printers, it is a printing system.

Furthermore, and as pointed out in the RAN, the claims do not require a single chassis. RAN at 11. Thus, the claims potentially cover functionality that is performed by an external (but connected) component. CCP tries to avoid the chassis argument, by saying that the claimed printer must be a single device. PO Brief at 22. This argument fails: the claims do not recite such a limitation, and even if they did, there is no requirement that a device must reside in a single chassis.

CCP also discusses the alleged commercial success of "placing the claimed functionality in a printer." PO Brief at 23. Samsung discusses the alleged evidence of commercial success below, but this argument is especially misplaced in the present analysis. CCP has not produced any evidence of commercial success for a printer that outputs a print data stream to another printer, as recited in claims 20, 59, and 64.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
95/001,398	07/16/2010	6684789	P-74637-US	5488

49443 7590 12/18/2013
Pearl Cohen Zedek Latzer, LLP
1500 Broadway
12th Floor
New York, NY 10036

EXAMINER
STEELMAN, MARY J

ART UNIT	PAPER NUMBER
3992	

MAIL DATE	DELIVERY MODE
12/18/2013	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

PTOL-90A (Rev. 04/07)

RECORD OF ORAL HEARING

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SAMSUNG ELECTRONICS CORP. LTD.
Requester

v.

CCP SYSTEMS AG

Appeal No. 2013-009045
Reexamination Control No. 95/001,398
Patent 6,684,789
Technology Center 3900

Oral Hearing Held: October 9, 2013

Before HOWARD B. BLANKENSHIP, STANLEY M. WEINBERG, and STACEY G. WHITE, *Administrative Patent Judges.*

APPEARANCES:

ON BEHALF OF THE APPELLANT:

GUY YONAY, ESQUIRE
Pearl Cohen Zedek Latzer LLP
1500 Broadway, 12th Floor
New York, New York 10036

ON BEHALF OF THE THIRD-PARTY REQUESTER:

DAVID L. McCOMBS, ESQUIRE
Haynes and Boone, LLP
2323 Victory Avenue, Suite 700
Dallas, Texas 75219

The above-entitled matter came on for hearing on Wednesday, October 9, 2013, commencing at 2:50 p.m., at the U.S. Patent and Trademark Office, 600 Dulany Street, Alexandria, Virginia.

P R O C E E D I N G S

\- - - - -

THE USHER: Calendar No. 23, 2013-009045, Mr. Yonay, Mr. Arjomand, Mr. McCombs, and Mr. O'Dell.

MR. YONAY: Your Honors, before we go on the record, I understand there will be some exhibits. May I provide a courtesy copy of a handout that we have. It only contains material that's in the record, and I know that it would not be entered, but it's just something to provide as a courtesy.

UNIDENTIFIED JUDGE: Yes.

MR. YONAY: Judge White, I will leave a copy here for you. I know you can't see it, but everything here is in the record.

JUDGE BLANKENSHIP: We're on the record, and we'll hear from the patent owner first, and you have 30 minutes, whenever you're ready to go.

MR. YONAY: Thank you. I'd like to reserve five minutes, please.

JUDGE BLANKENSHIP: All right.

MR. YONAY: Good afternoon. My name is Guy Yonay for patent owner CCP. The '789 patent -- I'll describe it briefly, the functionality right before I dive into the claims. Generally, what it allows is for a printer to become connected to a network in a very different way than it had been previously, and specifically, to make it a permanent two-way communication rather than merely the end point of the network, meaning it is no longer just an output point, but rather it can also provide input back to

the network. One example of how this would be applied in a business context is -- we provided an example like this in our brief. For example, invoices are printed out and, at a printer, as the invoices are printed, information is gathered from those printed invoices. So what's received is this print document, this invoice for printing. It is unpackaged and understood by the software, and scripts are assigned to various objects that are identified by this parsing operation. One such script may be, for example, collating or gathering certain types of information such as which salesmen made what sales. And then a report may be generated, for example, by sending that collected information to a printer on the network that's closest to where this salesperson sits. This ties into the claim in a very specific way. There are elements in the claim that are required that really make this invention work. And the three elements in Claim 1 that I'd like to focus on are the input print data stream, the parsing step, which is parsing the input print data stream for graphically representable objects, and the third element of the claim that I'd like to focus on is assigning a script to an object. And when we say "to an object," meaning to an object that's been identified by the parsing operation. This is all language directly from the claims -- from Claim 1. There is also a claim that I'll discuss, Claim 20, that says -- that recites a printer characterized in that it essentially performs those methods -- steps of Claim 1.

One more comment on the prosecution, before we get to the references: The Examiner in the underlying case in the original Notice of Allowance pointed to the combination of these elements being found

nowhere in the prior art. And I think we'll show today none of the references currently used in the rejections have this particular combination of steps.

I'd like to start by talking about the Interleaf reference. And when I say "Interleaf," I mean both the publication and the patent. They really refer both to the same system. And the Interleaf system is an active document system. It's a document you can create, view, and edit, etc., that includes active elements. So the content of the document is dynamic. It can be updated based on stock prices or the date or things like that. And the way that's done is the document includes both objects, as well as scripts, and specifically Lisp scripts. So Dutton has objects and scripts, and it manipulates these during the editing process. And eventually when the document goes to printing, the scripts are executed, the content is fixed, and then a print data stream is created. So the output of Interleaf's printing process is a print data stream, but the input is not a print data stream. The input -- what's manipulated by the Interleaf system is essentially the document file stream, which is how the Interleaf -- how the document refers to it. And there is certainly nothing to indicate that it's a print data stream that's being manipulated prior to creating the print output.

The second point about Interleaf that I'd like to discuss in addition to the input not being a print data stream is the parsing element. And specifically what we have in the claim is parsing the input print data stream for graphically representable objects. As far as we could tell what Samsung and the Examiner are pointing to in Interleaf, it is this quote here -- this is

the quote from Interleaf in Samsung's Brief at page 11, and the emphases on the object system and syntax checker are from the Samsung Brief, and this gap here between the sentences represents the ellipses in the paragraph in the quotation. And the first sentence here talks about the object system; that refers to the object -- the Interleaf document is an object-oriented format. And so the heart of I-6 is the object system and the class hierarchy in these things refer to the objects. The second sentence we see here is that the developer's environment includes various tools. And one of those tools is a syntax checker. But what we've shown and what you can see when we fill in the missing text is that the syntax checker actually refers to checking the syntax of the scripts. And specifically we have a new paragraph here that starts after the first sentence stating that Interleaf has an integrated Lisp system based on Common Lisp. And then we have the text that was quoted, "the Interleaf 6 developer's environment" -- in other words, the environment for developing these Lisp scripts includes an editor, listener, compiler, debugger, syntax checker, and other utilities. And the entire source at -- for the Lisp portion of the system. So what's clear from reading the full text of Interleaf here is that the syntax checker checks the syntax of the script. Now, why is this important? Because it switches the order of things, right? Because in our claim we have parsing and input print data stream, and then identifying objects and associating a script with an object. So you have first parsing, and then you associate a script, you assign a script, whereas the Lisp interpreter here, to the extent that it's a parser, parses a script and understands the script, meaning after it's already been assigned to an object.

This has nothing to do with parsing the input stream for objects. As I said, the Interleaf system already has -- an active document already has objects and scripts. You would not parse it to identify objects because they're already there. So the parser in Interleaf has nothing to do with the objects. It's to interpret the Lisp scripts.

One more comment on Interleaf. Were there questions about this? Okay. So the two main points I'd like to make today about Interleaf are: one, that its input is not a print data stream, and secondly that the document file stream, which is what comprises an Interleaf document, is not parsed. The only thing that we've seen in the rejection that is a parser is the syntax checker, which really is only applied on the script, not on the document file stream as a whole.

I'd also like to talk about Claim 20 with respect to Interleaf. It's kind of a high-level point. Claim 20, as I said, is a printer that performs the method of Claim 1. Interleaf is software on a PC. You can connect your PC to a printer, but that doesn't make the method run on the printer. The method is still running on the PC, and that doesn't change the functionality of the printer. So in other words, Claim 20 is not merely a system that does the method of Claim 1 that happens to have a printer. Claim 20 is pretty clear that what's being claimed is a printer that performs the method of Claim 1, with intervening Claim 17, but that's the idea -- that's the gist of Claim 20.

I'd like to move on to the IBM reference. And a word of introduction about the IBM reference: when we say "the IBM reference," what we're

referring to is a 400-page manual that describes printing functionality of the AS/400. The AS/400 is an IBM mid-range printer. It would typically be shared by many clients within a company, up to a few thousand typically might be sharing its resources. And this 400-page document deals with how you can format data for printing within the AS/400 environment. And the AS/400 runs many different types of applications, examples in the reference are invoicing, accounting, and things like that. And it sends output to many different types of printers, printers with a wide variety of functionality. And so it's not surprising that this 400-page document describes a number of different print formatting processes for different applications, for different printers. And, as we'll see, the rejection here by the Examiner was based on mixing and matching steps from different print processes improperly. The two processes in the AS/4 -- in the IBM reference that I'd like to talk about are the application formatting and the external formatting. Application formatting is where an application outputs a set of records that need to be printed. Really, the heart of the AS/400 is it's a big database engine, and an invoicing application may say, you know, "print these 15 records." Now, that application will have a DDS associated with a particular type of output. The DDS is a "data description specification." It's the way in which data is formatted for printing, and it's described by the application. I mention the DDS because that's what's been identified as the script in the rejection. So the DDS is analogous to the script in the rejection. And, as we see in the IBM reference, the DDS is applied to record output.

The second type of formatting is called "external formatting," and that refers to page definition and form definition. And, in fact, this reference is where IBM introduced PAGEDEF and FORMDEF. And the way that this works is an application spits out unformatted line data. It's a whole bunch of lines. And what the page definition does -- and this is a quote from the reference -- is it parses that line data and places it on a page. And that's where the Examiner found the parsing element. If we can have a figure up -- Figure 116 describes this pretty well. This is Figure 116 from the IBM reference that's been discussed in the briefs. And this is the full figure. What we see at the top here is the traditional print methodology, referred to as "application formatted," so the application outputs records as well as a DDS associated with those records. The spool applies the DDS to those records and that gets outputted to a printer.

What happens in the external formatting is the use of this page definition and form definition. The application here outputs unformatted line data, and you can -- if you look closely, you can see that schematically it's represented by the lines being kind of jumbled in this spool there. That's because they're unformatted; they're just lines. And the PSF/400, this box here, is the printing subsystem. What the PSF/400 does is it applies a page definition, and that's the parsing. It parses the line data and places it on a page, and we have a logical page here so you can see the lines are now neatly arrayed, neatly placed on the page. What happens at this stage here, the PSF/400 essentially takes that unformatted data and applies the page definition to create an AFP data stream, an "AFPDS." Within this box --

within the PSF/400, AFP is used. AFP is an object-oriented way of describing the data. So we took unformatted data; we've parsed it, placed it on a page, convert it to AFPDS. And then we can do other things with it. For example, this form definition may place an overlay onto the page. So you can kind of see that these boxes here and the logo are now placed onto the page. That's the form definition places the overlay onto the page.

And, finally, what happens at the end here is that this AFP data stream is converted to an appropriate format for the printer. In the case of an intelligent printer, IPDS is used, "IPDS" being an intelligent printer data stream. IPDS is the way the printer receives -- an IPDS printer may receive the data; if it's a less intelligent printer, it will use -- the AFP may be converted to SCS.

So the reason we're talking about these distinctions here is that the application formatting can use a DDS, which is the script referred to in the rejection, and the external formatting receives this unformatted data and has to parse it. So in the external application, in the traditional print methodology, we have the script. And to our understanding, this Figure 116 is the only place that Samsung has pointed to or that the Examiner has pointed to in which those two method steps we discussed supposedly happened together.

Now, I'd like to overlay onto this Samsung's limitation in its Brief at page -- page 5 of the Samsung Brief. So the first thing that we observe here is that the top portion is omitted from the figure. And essentially the text that -- this appears at pages 193 and 194, where the beginning of that text

starts by introducing the traditional methodology and the text segues to talking about the new external methodology. And by ignoring the traditional print methodology, what happens is all of that text is then applied to -- it's understood to apply to this external formatting, which is incorrect. And the two most egregious errors and the most material ones for this rejection are these two, are SCS supposedly being used as an input here, which is incorrect. And the second error is a DDS script supposedly being applied within the PSF/400 subsystem. I'll talk about those two mistakes.

We've said and the figure itself says that the application outputs unformatted line data. It is not SCS; it is not an input print data stream. In fact, in a few places in the IBM reference, it specifically says that the line -- line-formatted data is different from SCS. One of those, for example, is at page 213. And it talks about how SCS is different from the line data. An SCS is something that can go to a printer. It will typically have printer-control commands. And since the application here is sending unformatted data to the PSF/400, it will not include those printer commands. There is no need for them. What it sends out is this unformatted line data because it will be formatted by the PSF/400. So that's the first mistake. It does not receive an input print data stream.

The second mistake in applying the traditional print methodology text to the external formatting in the bottom of the figure is this DDS supposedly being applied. That is incorrect. A DDS is never applied in the external formatting. In fact, at page 37 of the IBM reference, the direct quote here is that the page definition is an alternative to DDS. Okay? So in a sense,

they're kind of competitors, right? I mean, DDS is one way to place information on a page, right? It's a data description specification. It will take the record and place it on a page. And the page definition is another way to that; it's a different way to do that. And it will take that -- there's nowhere that the DDS accepts unformatted line data. DDS accepts record data -- DDS really has nothing to do with this process here.

The one support we saw in Samsung's Brief for the DDS issue is a heading in the IBM reference that says, "Keywords From AFP Applications." And it says that certain DDS keywords can be used in AFP applications. Well, that's kind of reversing things again because an AFP application is an application that outputs AFP. In other words, an application that shortcuts this whole thing. If you don't need all this -- you can have an AFP application -- you can have an application that just outputs AFP and then you wouldn't need the parsing; you wouldn't need any of this stuff. And one of the tools that the AFP application can use are keywords from DDS. In other words, you can use DDS to create an AFP data stream, but not the reverse. You don't apply a DDS to something that's already an AFP data stream such as you would find within the PSF/400 printing subsystem.

And so, zooming out a little bit, what we have here is a -- you know, the Claim 1, as we said, is this series of steps. And it's almost as if each step has an input and an output where the input of the next step in the claim is the output of the previous one. So we have receiving an input print data stream, parsing the input print data stream for objects, storing the objects in object-

oriented format, and assigning a script to an object that's been stored in object-oriented format.

JUDGE WEINBERG: Is the word "assigning" in Claim 1?

MR. YONAY: There is.

JUDGE WEINBERG: I'm looking for it; I don't see it.

MR. YONAY: I'll put it up. Claim 1 is the original application here. The priority application was a German application, and so it's kind of in European format. You've got, in Claim 1, small Roman i, ii, iii, iv, v, and then you've got the "characterized," characterized in that graphically representable objects are stored in the memory in an object-oriented format to which at least one storage script is assigned, which is executed in the case as defined in the script. One of the things we tried to do before the Examiner was to recast these claims in a slightly more U.S.-friendly format. But yes, it's absolutely in there.

JUDGE WEINBERG: Counsel, didn't the Examiner already consider all the evidence of record and make a determination that's still obvious?

MR. YONAY: Well, the Examiner does state that -- the Examiner states a weighing was made. We don't know exactly how that weighing occurred, and I would point out there are several errors of legal analysis in the Examiner's argument. And I'm actually happy to turn to that because that was my next point, was to look at the Examiner's analysis of these factors. But I do think that it's worth noting that the secondary consideration's evidence here is of a different and stronger nature than what

you see in most cases. But let's turn to what the Examiner did because I agree with you.

JUDGE WEINBERG: What you're discussing is a method step, but it doesn't seem that this is a method step.

MR. YONAY: Well, it says "is assigned." A script is assigned. That to me means assigning a script. That's --

Another word about Claim 20 with respect to the IBM reference, as we said, Claim 20 recites a printer characterized in that -- and I recognize that I'm paraphrasing, but the gist of it is that the printer performs the steps of Method Claim 1. And what we have here is Figure 178 from the IBM reference showing the -- the AS/400 is the rectangle here on the left. It's got the various subsystems. It can be connected to a printer, and it can also be networked to additional printers. In the overlay here, what we see is, again Samsung's limitation, and this is from their comments on our response to the ACP. And what Samsung did here, and the Examiner also copied this into the Wren, put a box around the AS/400 and the printer. But that doesn't really solve the problem. It's kind of fiction here. Doesn't mean that the process is now performed on that printer. It also doesn't mean that the AS/400 becomes a printer, and it certainly would not have been obvious to take the massive processing power of this AS/400 that supports several thousand users and put it onto a printer. So Claim 20 is certainly nowhere anticipated or obvious, based on the IBM or the Interleaf references.

We talked briefly about the objective evidence here, and what's interesting in this case, I think, to point out is that the principal reference is

this IBM manual. And yet IBM itself licensed CCP software, the J Scribe software, for several million dollars. So, you know, we have the case where supposedly the licensee, IBM, would have supposedly already knew everything that CCP already had to offer in its patented software. And that just doesn't make sense. It wouldn't have been -- wouldn't have made sense for IBM to pay millions of dollars for some trivial variation on the AFP software that it already had -- on the AS/400 software it already had.

JUDGE WEINBERG: Did they license to patent it?

MR. YONAY: Pardon?

JUDGE WOOD: Did they license the patent itself?

MR. YONAY: They licensed the software that incorporates the patent, that practices the method of the patent. I don't believe there is anything below specifically saying that they took a license to the software. But when we talk about commercial success, what we talk about is the product, for example, right? The product is successful; the product generated praise. You don't see praise and -- for a patent. You see praise for a product.

JUDGE BLANKENSHIP: You're in your rebuttal time. You want to proceed or --

MR. YONAY: No, I'll reserve the rest of my time unless there are any more questions.

JUDGE BLANKENSHIP: Anything for now? All right.

MR. YONAY: Thank you.

MR. McCOMBS: Your Honors, I'm David McCombs here on behalf of the Requester Samsung. And Judge White, Judge Weinberg, and Judge Blankenship, thank you for your time today. I'd like to begin by putting this case in a bit of perspective. This is a situation where the Office had before it very strong anticipatory references with IBM and Interleaf. The intelligent printing functionality that is described in the '789 patent, this was just not new, and it's a situation here where the marketplace had already gone to migrating these enhanced printing functions away from the original document or the application on which it was created into a separate facility. And that's what the Examiner recognized. The Examiner was very thorough in her 160-plus page Action Closing Prosecution in which she very carefully mapped the claims -- the various features of the prior art references to these claims.

I'd like to address some of the points raised today by the patent owner, in particular looking at the IBM AS/400 reference. And this is a situation where when we see Figure 116, which is at page 194 of the IBM reference, that does show that there is an input data stream in which parsing is accomplished, and the graphically representable objects that are parsed are then -- scripts are assigned and executed to those. What we see really as a primary point is their questions about well, what is the input data stream and what kind of input data stream is it, and so forth, and what can be done with that. What IBM is very clear about is that what is input is -- there are many different printer formats that can be input. Many different data formats that can be input, and then the data stream is manipulated in various ways by the

system to produce an output. And at page 8 of IBM, what they talk about is print flexibility. And at page 8, they say, you know, there are many different print streams that are expected, and the goal is to provide for "any-to-any printing," in IBM's words. And so the whole point of the IBM system is to take input and convert it into an advanced function presentation (AFP) object-oriented format. And that AFP architecture that's used is -- then allows you to create and execute these enhanced printing functions. If you look at the IBM reference at page 24, it explains that there are many different input formats that can be used, and there are a number of different examples which are listed there. They range from simple line printer data on up to more advanced AFP, you know, data stream formats. And, again, what's going on with the AS/400 system is that it's all about formatting these data streams into ways that you can then provide for different ways of enhanced printing.

When we look at examples of what that data stream actually looks like, if you look at the IBM reference at page 208, there is an example of the input data stream that's being used for which, on Figure 116, we're going to be doing these other resources of creating page definitions and form definitions. And at page 208, again you can see what the input screen looks like there. It says right at the top, "The following is an example of line data produced by an application program." And you see that and it does have various fields and structures to it. And it's a stream that has a syntax that allows for parsing. And then what's described in this example, this is a case study that is used in the IBM reference for a hypothetical company called

Sun Seeds, and they show how invoices can be formatted and prepared for this invoice -- using this input data stream. And what they talk about, and you look at the example at page 213, there's an example of a page definition that's applied. And it's shown how the system, the AS/400, using this PSF/400 print facility, is how that page definition takes the data and parses it. And described at page 194 of the IBM reference, it says that the data is parsed into individual fields. When you look at the example of how that's done in this Sun Seeds case study, it's shown there on page 213, down at the bottom, one example being that the zip code field is parsed into a barcode object. So we're seeing here that the input into the system is clearly parsed. And then from there, as well, in what these page and form definitions do is that they also -- once the data is parsed into fields and the like, it enables you to provide various media-handling facilities. So, for example, what's shown is there's an overlay. And that overlay function is a script. And what that script does is it made do something like put a little graphic element such as the letterhead on the page of the invoice, or it may do something like put a graphic picture of a seed or a flower or something like that onto the print screen. And what creates some confusion here -- and I understand how this could happen -- the confusion is that the page and form definitions that are providing these kinds of scripts -- and, again, these are the same type of scripts that are described in the '789 patent at about column 6, lines 10 though 20, those same scripts are the same kinds of scripts that also are available through the data description specifications, or "DDS." And so you will have scripts that -- in DDS form that use the word, you know, "overlay,"

for example, or "rotate text" or "barcode" and other things. And those are all scripts provided and available through DDS (data description specifications) and also there are scripts similar to this that are -- that can be defined through page and form definitions. And the two of them can coexist and in fact do coexist. The IBM reference specifically talks about the fact that you can have many different ways that you do some of these enhanced printing functions. And if we look at page 37 --

JUDGE WHITE: Excuse me, counsel. I have a question.

MR. McCOMBS: Yes.

JUDGE WHITE: The rejections that are currently before us: Is the Examiner relying on scripts from these page and form definitions as the scripts in the claim? Or is the Examiner solely relying on DDS as the script?

MR. McCOMBS: Yes, Your Honor, the examples that the Examiner has been referring to are the DDS scripts. And if you look at the IBM reference, for example -- two places: if you look at page 145, there is an example of this case study involving the Super Sun Seeds invoice in which it's a DDS version. And, for example, it's showing there that you can provide scripts that do overlays. And you can see there, in one case, for example, there would be a flower object placed on the invoice. Or at the top there might be the letterhead for the Super Sun Seeds Company. And at the IBM reference, at page -- really the range is between pages about 127 through 134, but more particularly on page 132, there is an example overlay. And it's describing how the DDS specifications do provide overlays that would include graphics of this nature. The idea that there -- so to the extent

that there are keywords in DDS and keywords in page and form definitions, there can be some complementary use of those. What is described at IBM at page 37 is the idea that page definitions are optional, and if you don't specify a page definition, media-handling characteristics are taken from existing keywords. So the reference here is acknowledging that there's two different kinds of keywords that can be in an AFP-formatted data stream. And in some cases, if you decide to create a form definition and it uses something called overlay, that would override keywords that are already specified, for example, in this system. So it's discussing the idea that you can have different ways of providing scripts in the IBM system.

And that's also made clear, as well, if you look at pages -- about page 39, 40, and 41 of the IBM reference, it describes various enabling applications for tools that the AS/400 system uses to provide these kinds of enhanced features. And the tools that are described, one of them, for example, is called "PPFA," or "page printer formatting aid." And that PPFA tool is a tool that will provide page and form definitions and so on. Another tool that's described in that same listing is at the bottom of page 40 going up onto page 41, and it's called "report layout utility" (RLU) utility. And what that does is that builds DDS scripts. And so we have a situation where data that can be brought into the system as AFP, there can be -- these tools can all be used to provide different enhancements. And it's very similar in many ways to what's described in the '789 patent, if you look at how they claimed it in Claim 18. You can -- within their system, you can have -- which is shown on their figure -- there's a Box 8 which talks about using a display

and being able to perhaps enhance or edit or do other things to the data stream to further enhance it. The same kinds of tools are available in the IBM reference where you can make edits or changes or provide other enhancements or scripts to the data stream to suit your needs.

So what we have here is a situation where the DDS specifications that are relied upon by the Examiner are appropriate to show that scripting can be performed on the same data stream for which page and form definitions may also be used. And that's also shown when you look at IBM at page 25, where it actually shows how DDS keywords and page definitions can go together. Both of them are actually stored in the printer file and can be used to provide enhanced printing functions.

One of the things that I think was said, just to clarify, it would be a mistake to say that the DDS specifications can only be defined in the original application from -- in which the document was created because what IBM says is that DDS scripts can be externally defined, outside of the application. In fact, there's even discussion about how that's done in the AS/400 or in the AFP PSF/400 software, and it's done using different object code and source code than what would have been in the original application of which the document was created. And that's externally defined versus program defined.

Are there any further questions with regard to this before I move to the Interleaf reference? Okay.

So the issue on the Interleaf reference, which also the Examiner used for anticipation and, in some cases, obviousness here, that reference is

discussing the idea of using -- taking a document data stream and performing enhancements to the stream that would be in the form of parsing and scripting, and the use of that can be for printing. And I believe what the main concern or argument is by the patent owner with respect to Interleaf is that, "Well, that's all done in the document development environment, not in a printer environment." What you see, though, is that the Examiner was pretty well clear that, to the extent you're going to be taking this document data stream and you are going to be performing things on it that can include printing, then that is a print data stream. And it's also -- part of the reasoning there is it's even so with respect to the '789 patent. The '789 patent itself talks about taking this data stream and doing different operations on it, making modifications display, etc., and producing an output to an output device, preferably a printer but it doesn't require that it go to a printer. And so the -- trying to put these specific limitations into the data stream itself, or the definition of a data stream, doesn't change the fact that the same kinds of operations that are being performed in the IBM system and the Interleaf system and '789 system, they're all being done to this same data stream, parsing and scripting and output which can go to a printer or doesn't necessarily have to go to a printer.

I'll think I'll just point out, as well, one of the items that was raised here is that there was, in the Interleaf reference discussion of performing syntax analysis on a script as opposed to a -- that is, would be a script as opposed to actual data stream itself, that the Interleaf reference does describe that the data stream is parsed and at page 84 the Examiner recognized that if

the stream contains active objects, these become activated when the document's opened programmatically. That's describing what's going on with respect to parsing.

As far as scripts are concerned, they are likewise applied and executed, as recognized by the Examiner. It says in Interleaf that Lisp scripts can be attached to any Interleaf object. And then Interleaf gives examples of how these scripts, you know, can replace words in different languages. So there's a table at page 78 in the Interleaf reference that -- so if you are -- if you want to change something to a different foreign language, if you have the word "bullet," if you want that to appear in Italian, it would automatically change to say "marca." And so these show how scripts are applied and executed in IBM.

I think I'll reserve -- if you have any questions, I'll stop but -- oh, yes, just one other point with regard to Claim 18: Again, there was a picture in the briefing that the patent owner provided with respect to the fact that the Interleaf reference was really in another field of endeavor, having to do with printing as opposed to user interaction and document modification. And if you look at Claim 18 of the '789 patent, it specifically has all of the features that would go into user interaction and display and printing. So, really, the idea that the field of endeavor is limited is belied by the patent -- the '789 patent itself, which describes performing other kinds of enhancements to the data stream that may or may not involve printing or, as stated in Claim 1, sending an output, preferably to a printer.

So I'll stop there and reserve the rest of my time.

JUDGE BLANKENSHIP: All right. Any other questions for now?

JUDGE WEINBERG: No.

JUDGE BLANKENSHIP: All right, sir, thank you.

MR. YONAY: With regard the discussion about IBM and the DDS scripts and all that, we're still focusing on, you know, what -- what I'm still not seeing is the parsing and assigning a script in the same flow, in the same data flow. So if you have the -- you know, these figures focus on the page definition being the parsing, but you don't have a script that is assigned to an object that is the result of the parsing operation. And the example given here -- I think -- first of all, I think Judge White observed, you know, we're shifting the ground a little bit from what the Examiner said was the script which she pointed to DDS. Now it's DDS keywords in other environments or other applications. I'm not exactly clear what those are, but the example given here in the presentation is overlaying just like we saw in the form definition. What we saw, remember there was the page definition, then the form definition. And this overlay of the boxes and the logo would be done in a form definition, but that's not applied to an object that's identified from the parsing operation. It's just overlaid on the page.

Another thing that I haven't heard is -- you know, there's been lots of discussion about data streams, but neither for the Interleaf nor for this IBM reference is the input a print data stream. And I think the discussion was very precise that Interleaf manipulated a document data stream. But that's not the same thing as a print data stream. And if there's any ambiguity at all -- and we don't think there is about what a print data stream is. A print data

stream is not defined by where it goes, that it goes to a printer or doesn't go to a printer. What defines a print data stream as opposed to a generic data stream is that it's formatted for the control of a printer. And where it goes is irrelevant. The Examiner recognized that but there must be some patent -- some weight given to the word "print" in this context. And a print data stream differs from any generic data stream in that it is formatted for the control of a printer. And if there is any doubt at all --

JUDGE WHITE: Excuse me, counsel.

MR. YONAY: Yes.

JUDGE WHITE: Can you point me to anything in the specification that defines the print data stream?

MR. YONAY: It's not explicitly defined. But it's really discussed throughout what a print data stream is. One example is a particular type of print data stream. And that is a -- for example, a page description layout is one type of print data stream. For example, at column 1, lines 13 to 16 or so, it says there's a driver that converts information about the graphic objects to be outputted, for example, text storage information, into the respective PDL (page description language) suitable for the printer use so that the latter can be controlled directly. In other words, the PDL is one type of print data stream, and what it does is it controls the printer. That's true of any print data stream is it controls the printer. There are other instances. For example, when referring to the output print data stream, for example, in column 3, lines 13 to 15 or so, says that the objects are transformed into a format for the control of an output device, preferably a printer. So what

makes a print data stream is that it controls a printer. Or it is -- it is formatted for controlling a printer. Whether or not it goes to a printer or not is not what defines it.

JUDGE WHITE: Let me make sure that I'm understanding something correctly from the patent because I thought the patent defined "postscript" as a type of PDL, which is really just a document format. So the document format such as postscript, you're saying that that is for control of a printer and not just a document format?

MR. YONAY: Postscript is a type of -- is a page description language. It is a type of print data stream.

JUDGE WHITE: Okay.

MR. YONAY: I see I think my time is up. Thank you very much.

JUDGE BLANKENSHIP: You've got 30 more seconds, if you like.

MR. YONAY: All right. Just a quick word about the -- there are some email claims where we have a number of claims that talk about that the script is capable of sending and receiving emails. And this board here shows the fundamental difference between our patent claim and the '705 patent, which was cited by the Examiner. In our patent claim, it's the script that triggers the sending or receiving of an email. So you have the print data stream; it's parsed, etc. etc, script is assigned, and the execution of that script triggers the email. In the '705 patent, it's the causal opposite. What we have is an email is received, then a script is triggered and the document is achieved. It's really the causal opposite of what we have in the claim.

JUDGE BLANKENSHIP: Are there any other questions for now?

JUDGE WHITE: One question on that point. Did you respond to the section of the '705 patent that cited I think it's the bottom of column 1 to the top of column 2 that references a printer -- a print server, I believe it is, that's sending that particular email? I think that's cited in -- let's see --

MR. YONAY: Bottom of column 1, top of 2 -- did you say bottom of column 1 and top of 2?

JUDGE WHITE: Bottom of column 1, it's -- it starts, "Messaging is also used in the" --

MR. YONAY: Well, what I read here is that it says that the printer server receives an email.

JUDGE WHITE: Yes.

MR. YONAY: And it prints a document at the appropriate printer. So, for example, an email being included in an attachment as a pdf -- the attachment to the email might be a pdf, and the printer server then sends that pdf to the printer to be printed. So still what we have is an email that triggers a script within the print server. What we don't have is a script that triggers the email. That's -- okay. Thank you. Thanks very much.

JUDGE BLANKENSHIP: All right.

MR. McCOMBS: I have just quick points. First of all, with respect to the idea that parsing on the same object for which a script is -- would then be applied: The '789 patent's directed to the idea of assigning a script to at least one graphically representable object. And it can be, in some cases, some examples in the '789 patent are directed to the whole document or the whole page, like folding. There's different kinds of scripts that do things

like folded in a special way, stapled, do these sorts of things. These same kinds of scripting functions are -- to which a -- that may have been a document that may have been created using a page definition or form definition or which was somehow parsed and converted to AFP format by page and form definitions, the different kinds -- as those I've said, regarding page 37 of the reference, the IBM reference, there may be scripts or keywords that would have been created in DDS that would also be usable and apply to that same data stream to do things like folding or so forth. And it's also possible that later, once the data stream has been reformatted using page and form definitions, there's nothing in the IBM reference that says that you wouldn't be able to use the RLU tool to then create further enhancements, like you decide you want to different scripts, such as, you know, sending it a certain place or putting it into a different printer tray, and so on.

So there's different ways in which many of these tools can all be complementary. And I think the Examiner recognized that the kind of scripts that can be applied using DDS or using page and form definitions are all complementary and would all be applied to the data stream in the AFP system.

The other point I wanted to make is, you know, just with respect to the -- you know, the issues with regard to the emails and the '705 patent. I think there the Examiner was comfortable that the '705 patent was applicable and taught those features. If we go to the Action Closing Prosecution on that, the '705 patent, you know, would teach scripts that are -- teaching scripts for

receiving emails to Interleaf, patent teaches scripts for sending emails, and that was sufficient for the Examiner in combination to be satisfied that the '705 was in the right away of technology as applied to the whole idea of document publishing to feel that the combination was appropriate.

With regard to any other arguments that were made with regard to commercial success, I had not addressed that. And the Examiner was clear that all of the commercial success issues that were raised were in fact considered and the real issue there was just that there wasn't a sufficient nexus or linkage of those commercial success arguments to the specific claims in the patent. And, of course, since most of the rejections were anticipation, they wouldn't apply anyway. But to the extent that there would be specific claims for which obviousness was at issue, there was really no linkage between any actual commercial success in those claims. And the idea that there was a license of the software is not sufficient to show a nexus. The license of the software included object code and other things that had nothing to do with the patent. And there was no reference that we're aware of in any license to a license to the patent.

Are there any further questions? Thank you, Your Honors.

JUDGE BLANKENSHIP: No questions.

MR. YONAY: I'm sorry, before we leave and adjourn, Your Honor, I just wanted to make a note. I haven't reviewed this presentation fully, but to the extent that it has hanging material that's not in the briefs, obviously we object.

JUDGE BLANKENSHIP: Right. It won't form part of our decision.

MR. YONAY: Thank you.

JUDGE BLANKENSHIP: All right, sirs. Thank you, counsel. The case is submitted.

(Whereupon, the proceedings, at 3:47 p.m., were concluded.)

PATENT OWNER:

PEARL COHEN ZEDEK LATZER, LLP
1500 Broadway
12th Floor
New York, NY 10036

THIRD PARTY REQUESTER:

HAYNES AND BOONE LLP IP SECTION
2323 Victory Avenue
Suite 700
Dallas, TX 75219

UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT
and
THE UNITED STATES PATENT AND TRADEMARK OFFICE

Samsung Electronics Corp. Ltd.,	§	
	§	
Appellant,	§	
	§	
v.	§	
	§	
Acting Director,	§	
United States Patent and Trademark Office,	§	**NOTICE OF APPEAL**
	§	
Appellee,	§	
and	§	In re U.S. Patent No. 6,684,789 B2
	§	*Inter Partes* Reexamination No.: 95/001,398
CCP Systems AG,	§	PTAB Appeal No.: 2013-009045
Appellee.	§	
	§	

Notice is hereby given that Samsung Electronics Corp. Ltd. appeals to the United States Court of Appeals for the Federal Circuit from the Decision on Appeal of the Patent Trial and Appeal Board, United States Patent and Trademark Office, entered on December 20, 2013.

No fees are believed to be due to the United States Patent and Trademark Office in connection with this filing, but authorization is hereby given for any required fees to be charged to Deposit Account 08-1394.

Respectfully submitted this 19th day of February, 2014.

/s/ David L. McCombs
David L. McCombs
USPTO Reg. No. 32,271
HAYNES AND BOONE, LLP
2323 Victory Avenue, Suite 700
Dallas, Texas 75219
Telephone: 214/651-5533
Facsimile: 214/200-0853

CERTIFICATE OF SERVICE

Pursuant to M.P.E.P. § 2683 and 37 C.F.R. §§ 1.248 and 1.983(b)(3), the undersigned attorney for Appellants certifies that a copy of the NOTICE OF APPEAL was served, via United States Postal Service First Class Mail, on February 19, 2014 on the counsel for Patent Owner at the following address:

Pearl Cohen Zedek Latzer, LLP
1500 Broadway, 12th Floor
New York, NY 10036

/s/ David L. McCombs
David L. McCombs

D-2240962_2

Docket No.: 714092800100

UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

CCP Systems AG
Patent Owner or Appellant

v.

Patent Trial and Appeal Board
United States Patent and Trademark Office

NOTICE OF APPEAL

In re U.S. Patent No.:
6,684,789

Inter Partes Reexamination No.
95/001,398

RECEIVED
2014 FEB 19 PM 3:24
US COURT OF APPEALS
FEDERAL CIRCUIT

Pursuant to 37 C.F.R. § 90.2, Appellant CCP Systems AG hereby notifies the Court of its Notice of Appeal with the United States Patent and Trademark Office from the decision of the Patent Trial and Appeal Board dated December 20, 2013. The requisite copies of the Notice of Appeal as well as the requisite fee are submitted herewith.

Respectfully submitted,

Date: February 19, 2014

By:

Mehran Arjomand
Registration No. 48,231
Morrison & Foerster LLP
707 Wilshire Boulevard
Los Angeles, California 90017
(213) 892-5630

Attorney for Appellant

la-1238874

Reexamination Control No.: 95/001,398 Docket No.: 714092800100

CERTIFICATE OF SERVICE

I am employed in Washington, District of Columbia. I am over the age of 18 and not a party to the within action; my business address is: Morrison & Foerster LLP, 2000 Pennsylvania Avenue, NW, Suite 6000, Washington, D.C. 20006-1888. I hereby certify that copies of the attached

NOTICE OF APPEAL

was dispatched to the Court and served on the attorneys of record for The United States Patent Office on February 19, 2014 as follows:

THREE COPIES TO THE COURT VIA HAND DELIVERY

Clerk of the Court
U.S. Court of Appeals for the Federal Circuit
717 Madison Place, NW
Washington, D.C. 20439

COPIES SERVED UPON COUNSEL VIA UNITED STATES EXPRESS MAIL

The Office of the General Counsel
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, Virginia 22313-1450

Patent Trial and Appeal Board
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, Virginia 22313-1450

Haynes and Boone, LLP
IP Section
2323 Victory Avenue, Suite 700
Dallas, Texas 75219

I declare under penalty of perjury that the foregoing is true and correct.

Executed on February 19, 2014.

Cathy Woods

la-1239477